ABSTRACT :

Modern times need modern ways of production. Robots have been created for this purpose.
They can perform accurate movements in a short time. Their capability of speed and accurary make them ideal for many of the current industrial processes and for obtaining good economic results .

This project deals with a robot used for pharmaceutical purposes.
This robot can perform different activities, such as pippeting or obtaining the weight of a sample, all that with incredible precision and with no human intervention .
Any task may be performed completely automatically, so human errors are not possible .
Moreover, it may occur that the equipment needs to handle hazardous substances, so a human being might be affected by them if he had to handle those substances by himself .

The contents of this report are those ones related to the detailed description of the characteristics of each part of the equipment, as well as the programming of the robot and those devices used by it . Several programs have been developed to show robot operation .

The way to assemble all the equipment and connecting each part is also explained thoroughly, so new placement of the equipment will not require much effort .
Because of that, a User's Manual has also been developed and it is enclosed in this report .

Finally, it is to say that this current report is intended to be a complete reference guide to all the coming users of the equipment, such as teachers and pupils .

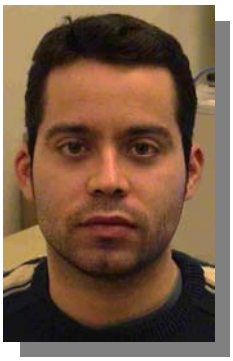
TABLE OF CONTENTS

About the authors 3

A Introduction 4-10
A1 What's an automated system? 5
A1.1 The need of automation & robotics 5
A2 But, what's a robot? 6
A2.1 Type of robots 6-7
A2.2 History 8-9
A2.3 Industrial aplications 10

B Our project 11-89
B1 Introduction 12
B2 Robot (Jaguar) 13
B2.1 Introduction 13
B2.2 Technical data 13-14
B2.3 Wire connection 15
B2.4 Commands and variables summary for the robot 16-17
B2.5 Re-teaching again the positions when a station is removed 18-19
B3 The controller 20
B3.1 Introduction 20
B3.2 Specifications 20
B3.3 Wire connections 21
B4 Stations 22
B4.1 Introduction 22
B4.2 Original position of each station in the plate 22
B4.3 Detailed information on each station 23-58
B4.4 General demostration program using all the available stations 59-60
B4.5 General view of the robot and the stations 61
B5 Power and Event Controller (PEC) 62
B5.1 Introduction 62-64
B6 Software 65
B6.1 Introduction 65
B6.2 Software terms 66
B6.3 Zymate files 67
B6.4 On-line / Off-line menus 68
B6.5 Dictionary 69
B6.6 Getting started 70-77
B6.7 Loop and conditional commands 78
B6.7.1 Loop commands 78-80
B6.7.2 Time commands (pausing) 81
B6.7.3 Examples 82-85
B6.8 Re-teaching position when a station is placed into a different location 86-87
B7 Problems and solutions 88
B8 Conclusions 89
B9 References 90

ABOUT THE AUTHORS OF THIS REPORT



Name : Jorge Montero López
Career : Industrial Engineering on Industrial Electronics
Home university : Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa (EUETIT) – Terrassa - Spain
University in Sweden : Royal Institute of Technology Kungl. Tekniska Högskolan - KTH (Campus syd – Södertalje – Sweden)

Name : Jordi Rodó Sarró
Career : Industrial Engineering on Industrial Electronics
Home university : Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa (EUETIT) – Terrassa - Spain
University in Sweden : Royal Institute of Technology Kungl. Tekniska Högskolan - KTH (Campus syd – Södertalje – Sweden)

Teachers :

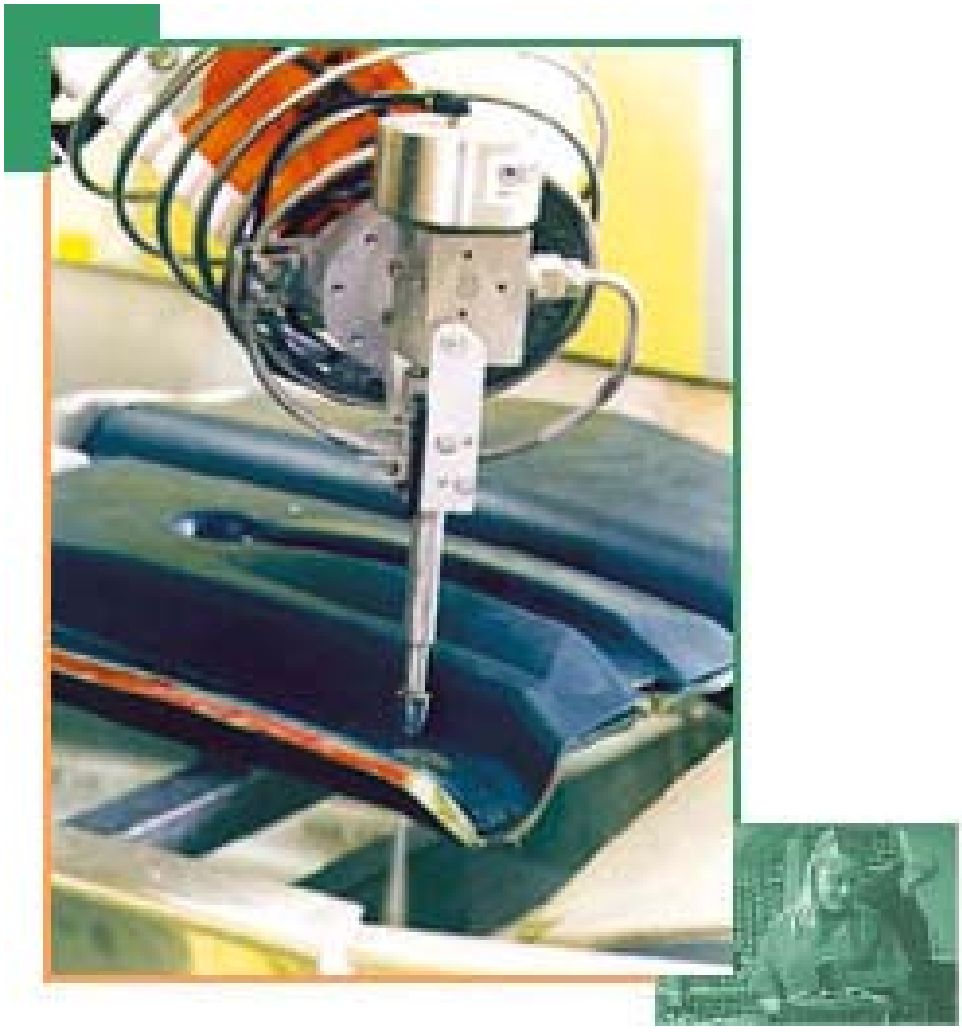
In Sweden :

Name : Mr. Christer Albinsson
Subject : Microprocessors, Real Time Systems, Embedded systems & Digital Design .
University : Royal Institute of Technology Kungl. Tekniska Högskolan - KTH Södertälje, Sweden

In Spain :

Mr. Albert Masip (Jorge Montero's teacher) : Department of Automation & Industrial Computering
Mr. Emiliano Aldabas (Jordi Rodó 's teacher) : Department of Digital Electronics

A . INTRODUCTION



(© ABB Automation)

A. INTRODUCTION

1. What's an automated system ?

The meaning of this word can be described as follows

Automation:

1. The implementation of processes by automatic means.
2. The investigation, design, development, and application of methods of rendering processes automatic, self-moving, or self-controlling.
3. The conversion of a procedure, a process, or equipment to automatic operation.

1.1 The need of automation & robotics

As the current economic system forces companies to automate their industrial processes in order to produce better and faster than their rivals, automating and robotics are needed. Companies, therefore, can substitute human operators with autonomous machines.
All things which need done are completed and/or supplemented by automation, reducing money, errors, danger, and most effectively time .

So there are not only economic reasons for automating, but also for safety, reliability and pecision . This current project is an example : this robot was made for working in a chemical industry. Sometimes workers have to handle hazardous substances and might get serious injuries. So the best option is to place a robot to perform those dangerous activities.

Other dangerous activities in which robots are used are, for instance, arc welding,foundries, painting and coating …

Pict. 1 : Robotic application for an automated task (© ABB automation)

2. But, what is a robot ?

As the topic discussed in this project is robotics, it's absolutely necessary to know what this word means :

Robot :

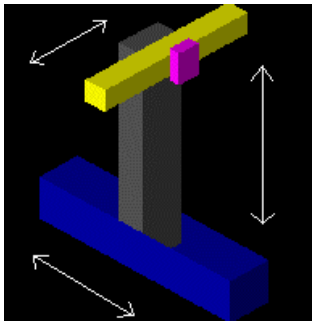
> A **robot** is a machine designed to execute one or more tasks repeatedly, with speed and precision.

2.1 Types of robots :

There are as many different types of robots as there are tasks for them to perform.

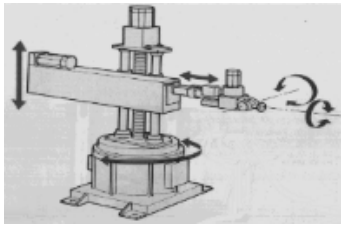
A robot can be controlled by a human operator, sometimes from a great distance.
But most robots are controlled by computer, and are classified into different categories: cartesian, cylindrical, polar, articulated-arm and SCARA (Selective Compliance Assembly Robot Arm) robots .

1. **Cartesian** co-ordinate configuration robots :



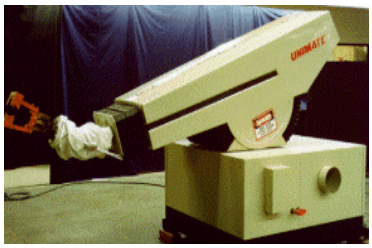
Three directions X, Y and Z are specified.
Co-ordinate directions are orthogonal (at right angles) to each other.
These robots move in three directions, in translation, at right angles to each other.

2. **Cylindrical** co-ordinate configuration robots :



The cylindrical configuration, as shown in the figure to the left, uses a vertical column and a slide that can be moved up or down along the column.
The robot arm is attached to the slide so that it can be moved radially with respect to the column.
By rotating the column, the robot is capable of achieving a work space that is similar to a cylinder.

3. **Polar** configuration robots :



Two of the joints move in rotation and the third moves in translation. The robot is capable of moving its arm within a spherical space. The figure to the left shows a robot with polar configuration.

4. **Articulated-arm** configuration robots :

All the movements of joints or arm members are rotary. This configuration is similar to that of the human arm. It consists of two straight components, corresponding to the human forearm and upper arm, mounted on a vertical pedestal. These components are connected by two rotary joints corresponding to the shoulder and elbow.
A wrist is attached to the end of the forearm, providing several additional joints.

5. **SCARA** configuration robots :

This configuration is a special version of the Articulated-arm configuration. The shoulder and elbow joints rotate about vertical axes. The SCARA configuration provides substantial rigidity for the robot in the vertical direction, but compliance in the horizontal plane. This makes it ideal for many assembly tasks.

2.2 History :

Robots are sometimes grouped according to the time frame in which they were first widely used. First-generation robots date from the 1970s and consist of stationary,nonprogrammable, electromechanical devices without sensors.
Second-generation robots were developed in the 1980s and can contain sensors and programmable controllers.

Third-generation robots were developed between approximately 1990 and the present. These machines can be stationary or mobile, autonomous or insect type, with sophisticated programming, speech recognition and/or synthesis, and other advanced features.
Fourth-generation robots are in the research-and-development phase, and include features such as artificial intelligence, self-replication, self assembly, and nanoscale size (physical dimensions on the order of nanometers).

Pict. 2 : a cartesian robot

Pict.3 : an arm robot

Pict. 4 : SCARA & cartesian robots
(© for pict. 2 to 4 : ABB automation and Seiko)

2.3 Industrial applications

These are the fields in which robots are used :

- ArcWelding
- Automotive
- Cutting and Finishing
- Consumer goods
- Gluing and Sealing
- Foundry
- Reconditioning
- Material handling
- General Manufacturing
- Packaging
- Metal fabrication
- Painting and Coating
- Plastics
- Spotwelding
- Chemical applications …

Pict. 5 : application for bottling (© ABB automation)

As shown above, several industrial activities can be performed with the help of robots.
They contribute to achieve excellent results in precision and speed, and that means, in the end, good economic results .

B. OUR PROJECT

(© Zymark Corporation)

B.OUR PROJECT

1.INTRODUCTION

Nowadays industries of all kinds have included automation in nearly all their productive activities in order to achieve better results on precision, time, safety…
This current machine is a robot made exclusively for chemical and pharmaceutical applications by Zymark Corporation (USA) .

Zymark Corporation (www.zymark.com) is a premier provider of products,technologies and services for laboratory applications in the rapidly expanding life sciences marketplace. Zymark is focused exclusively on laboratory automation and robotic solutions. The company's products span the full range of user needs from low-cost, single-function workstations to today's most sophisticated modular robotic systems.

This robot and its surrounding stations were bought by Astra-Zeneca, one of the Swedish leading pharmaceutical companies,and it was donated some years later to Kungl Tekniska Högskolan (Campus Telge, Södertälje, Sweden) .

We, Jorge Montero López and Jordi Rodó Sarró , were at the current moment of writing this report,Electronic Engineering students at Universitat Politècnica de Catalunya and we were taking our final exam work in Sweden, in the framework of the European Erasmus-Socrates exchange programme .
Our work has consisted in studying thorougly, documenting, re-assembling and re-programming again the cylindrical robot and all the different stations .

Some problems we have had are mainly those ones related to pressure-air based stations and finding out the correct wire connections between the controller, the robot and the power and event controller, since all the equipment had been taken to pieces when it was delivered to university .

2.ROBOT (Jaguar):

****

2.1 Introduction

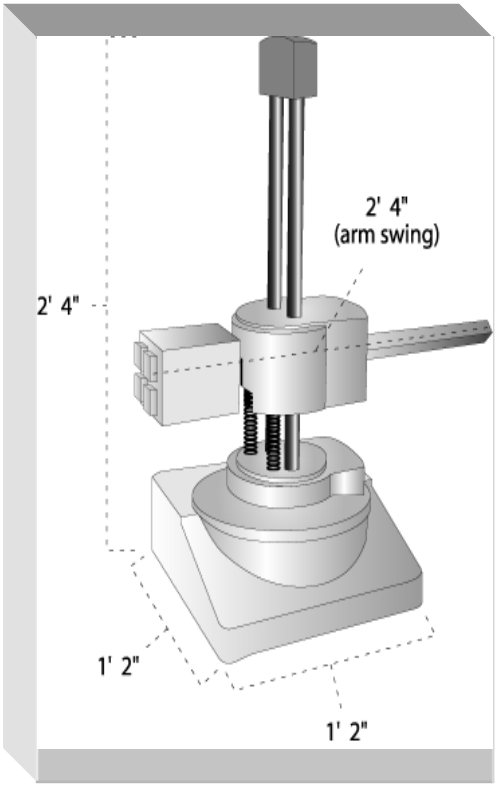
This robot provides the fast, efficient sample handling available for laboratory automation systems.

The Zymate robot is a fully programmable laboratory robot . Its' movements, speed and tactile sensing capability can be optimized for the requirements of the application. The robot's fully independent reach and vertical axes provide flexibility and motion control .

It incorporates interchangeable hands to carry various containers and samples, and to perform functions such as gripping and pipetting.
These hands are optimized for a specific purpose and several hands may be used throughout a procedure.Tactile sensing capability is included in the robot hand and all robot axes to verify the successful completion of robot functions .
The use of both digital encoders and potentiometers provides excellent positional linearity, resolution, and absolute positioning.

2.2 Technical data

Look at this technical information to learn more about the robot and its dimensions, performance & material specifications and its operating characteristics.



Pict. 6 : dimension measures

Dimensions:

Height: 71.1 cm
Width: 35.6 cm
Depth: 35.6 cm
Arm Swing Length: 71.1 cm
Weight : 39.2 kg

General Specifications:

Degrees of Freedom (Base & Wrist) : 4
Degrees of Freedom (Hands): 1 or 2 additional
Axis Positioning: Servo motor driven / AccuTrak feedback
Lifting Capacity: 1.4 kg
Hands : Auto–changeable
Operating Weight with Core Plate : 22.4 kg
Shipping Weight : 66.4 kg

Vertical Spring Holding Capacity : Standard GP Hand holding a sample container of no more than 700 grams — When power to the robot is turned off, or when collision detection or thermal protection occurs in the vertical axis, the vertical height of the robot will seek a neutral point as long as the sample and container weight does not exceed this weight. If this weight is exceeded, the arm settles to a vertical height of zero.

Performance Specifications :

Repeatability (same direction) :
Rotary @ 27 in radius : ± 0.5 mm max
Vertical : ± 0.25 mm max
Reach : ± 0.25 mm max
Wrist : ± 0.5 mm max
Repeatability (opposite direction) :

Rotary @ 27 in radius : ± 0.00 to 2.0 mm overlap max
Vertical : ± 0.5 mm max
Reach : ± 0.5 mm max
Wrist @ 6 in radius : ± 1.0 mm max
Warm - up Error Rotary Axis @ 27 in radius :
Zero (0) degrees : < 0.5 mm
Span (360) degrees : None with AccuTrak
Temperature Error : Rotary Axis @ 27 in radius:
Zero (0) degrees : < 0.5 mm
Span (360) degrees : 1.0 mm /10° C typical

Material Specifications :

Drive Cables :
Material: HT SS : 7 bundles, 19 strands/bundle, nylon jacket
Rating :160 lbs (used at 15 lbs typical)
Bearings:
Type : ball or tapered roller bearing assemblies
Lubrication : permanently lubricated motors
Motors:
Type: permanent magnet, d-c, brush-type, servo motor
Gear Reduction : integral gear reduction housing
Rotary, Vertical and Reach :19.7:1
Wrist : 187:1
Encoders:
Rotary: AccuTrak - optical encoder with 36 poles
Vertical and Reach: AccuTrak - magnetic encoder with 6 poles
Potentiometers:
Base Axis Position:10 turn, hybrid element, servo mount (2k or 10k)
Wrist Position: 1 turn, conductive plastic, servo mount (2k)
Arm: Stainless Steel, abrasive ground
Posts: Steel, centerless ground, oxide passivated
Lubrication (arm & posts):SAE 30 automotive–type engine oil

Operating Characteristics :

Axis Displacement :
Rotary: 376° ± 4
Vertical: 34.0 cm ± 0.5 cm
Reach: 32.0 cm ± 0.5 cm
Wrist: 550° ± 1° (-185° to 365°)
Reach Radius (at maximum): 27.0" (68 cm) typical (with general purpose hand)
Transfer Speed (time to transfer a container from one position to a second position); includes a rotary move of 90°, four vertical moves, two grip moves, and one reach move: 3 seconds

Axis Operating Forces (typical force vs. position error):
Rotary: 0.4 oz/0.1° (at 27" radius)
Vertical: UP - 1.3 kg / 2 mm error
DOWN - 0.4 kg/1 mm error
Reach: 0.4 kg/1 mm error
Axis Stall Forces (typical maximum force required to stall an axis):
Rotary : 1.4 kg (at 27" radius)
Vertical: UP 9 kg
DOWN : 14 kg
Reach: 4 kg
Wrist : 10 in-lbs
Cycle Time: 1.5 sec (forming a box moving 2.54 cm up, 30.5 cm across, 2.54 cm down and back)

→

Dwell Time : 5 ms (XP look ahead)
Finger Travel : 1 5/15" (3.33 cm)
Payload :1.36 kg continuous, 20kg transient

Maximum Speed :

Rotary : 298 cm/sec rotary 0.24 sec 90°
Reach : 99.5 cm/sec
Vertical : 73.3 cm/sec
Work Envelope : 9 m3
Cross Sectional Work Envelope : 0.27 m2

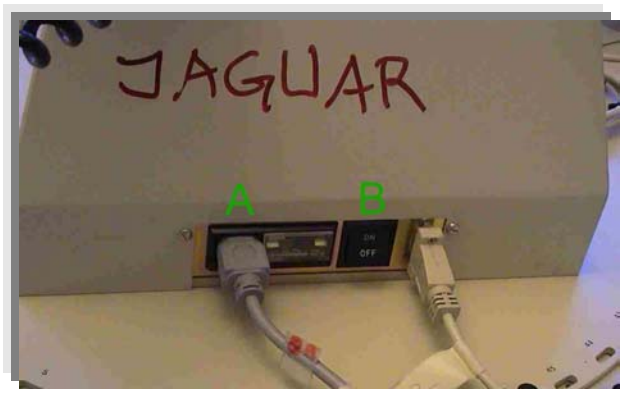
Environmental Specifications :

Main Voltage : 100, 120, 220, 240 VAC ±10%
Main Frequency : 48 - 62 Hz
Power Consumption : 180 VA max
Operating Temperature : +10° C to +30° C
Relative Humidity : 10 - 90%
Safety : Not to be used where explosive concentrations of flammable vapors can accumulate
Corrosive Materials : Standard Robot is not intended to handle corrosive materials
Mounting : Upright on flat, horizontal surface

2.3 Wire connection

Once the robot is placed onto a flat surface it must be connected to a power source (electric current).

Moreover,the robot must be connected to another device in order to get all the orders that it has to perform, since it is only a mechanic structure. This device is the **controller** . Picture 7 shows the conections to be performed between the controller and the robot :



A : to electric power
B : to the controller

Pict. 7 : wire connection for the robot

2.4 Commands and variables summary for the **robot** :

Next commands are used exclusively to make the robot move in different ways as to shown below :

Commands for the **hands** :

get.hand.a : when executed, the robot moves and gets in front of the hand. Then, by performing a linear movement onwards, attaches hand A (fingers).
get.hand.k : the same procedure as hand A. Hand K (pippet) is attached .
get.istd.hand : internal standard hand (liquid injection) is attached .
park.hand : the robot gets in front of the frame and by a linear movement parks the hand in its former place. This command is valid for any type of hand (A, K & istd) .

Commands for the **racks & stations** :

These commands are used to perform movements over certain stations : rack 2, balance, vortex, capper and crimp capper. Some must compulsory be executed, such as aspirate.2ml.tip (which needs move.over.rack.2 before) .

move.over.xxx commands :

move.over.rack.2 : performed when needing to place any hand over this rack .
move.over.balance : this is a sub-routine included, for example, in *put.into.balance* and performs a movement over the balance. The hand stops just above the balance door .
move.over.capper : sub-routine which places a test tube above the capper
move.over.crimp.capper : to place a tube on the crimp capping station
move.over.vortex : allows the hand to be placed above the vortex hole in where the test tube is to be put (sub-routine included in put.into.vortex)

put.into.xxx commands :

put.into.balance : includes the instruction called move.over.balance
put.into.capper : includes the instruction called move.over.capper
put.into.centrifuge : performs the movement to place a test tube into this station
put.into.crimp.capper : includes the instruction called move.over.crimp.capper
put.into.evaporator : the robot puts a test tube into the evaporator
put.into.rack.1
put.into.rack.2 : includes the instruction called move.over.rack.2
put.into.tumble.mixer
put.into.vortex : includes the instruction called move.over.vortex

get.from.xxx commands :

get.from.rack.1 : takes a test tube from this rack
get.from.rack.2 : takes a test tube from this rack
get.from.centrifuge : takes a test tube from the rack placed on the tumble mixer station
get.from.balance : performs the movement to get the test tube from the balance
get.from.capper : once a tube has been capped/uncapped, it is taken by the hand A
get.from.crimp capper : to take a capped tube from this station
get.from.evaporator : takes a tube from the evaporation station
get.from.tumble.mixer
get.from.vortex : takes a tube from the vortex station
get.from.vortex.nunc : (not used. Imminent crash possible)

2.5 Reteaching again the positions when a station is removed :

Once a station(*) surronding the robot has been removed and assembled again **in a different location** than when mounted for the first time , the positions concerning that station must be retaught 📄 .
If the station is reassembled in the same place than before only **fine-tuning** is needed ✎ .

(*) : a station (PyStation ©) is one of the elements that surround the robot, such as the balance or the racks .

⏭ Observation : two complete examples have been done to show how to perform this operation .
Please go to chapter 6.8 for a complete walkthrough .

📄 **Teaching** positions when a station is removed :

1. Print the PYSECTION.INFO program for the current section.The listing of the positions to be taught may be found in the "Rack Definitions" portion of the printout .
   Observation :for the Centrifuge station go to Direct Control. From Direct Control type DEL F:AT.ROTOR. This deletes this position from the dictionary in preparation for reteaching the position at the Cntrifuge's new location.
2. Prepare to reteach the rack positions by entering the Bench and module setup screen for the robot .
3. Press "9" for the Define Rack option and reteach the rack coordinates .

   Observation :for HAND.A press 9 for the Define Rack option and reteach Z:CLEAR.HAND.A (or Z:CLEAR.HAND.2).
   Observation II : for the Centrifuge station press 9 and follow the on-screen prompts to create the new position, F:AT.ROTOR.

   3.1 . When the screen says "The rack exists. Do you wish to Change (C) or move it ? Press C .
   3.2 . Single positions exist as two-position racks, with both positions occupying identical locations in a 2 row x 1 column matrix .
   3.3 . While teaching the rack positions, gently tap each side of the robot wrist box to confirm the hand's positioning .

4. Perform step 3 on all rack definitions taught to the system as a "rack ".
   Observation :for HAND.A reteach absolute position Z:AT.HAND.A (or Z:AT.HAND.A.2). This position is defined with the wrist coupling against the back of the hand (actually against the back of the monument since the monument should be used when teaching this position) . When complete, remove the monument and place the hand in its parking position .

5. When all rack definitions have been retaught, go to Direct Control, type **Test** and press Enter.

6. Verify the correct performance and repeat this process if not .

Fine-tuning positions :

When a station has been removed from its original location, its positions will need to be redefined as the bench is configured. After you Pyappend (see chapter 6 for further information) the software, the Setup program asks if the station is mounted on a Pyplate. Press **N** , print out PYSECTION.INFO, and redefine all station absolute and rack positions. (Relative positions should not require redefining, and the need for any incremental position changes is not likely to be apparent at this time).
When a rack (**) definition is being changed, the robot moves to the three previously defined corners for fine-tuning. If the station orientation differs greatly from the orientation when mounted on the PyPlate, make sure that nothing is in the way as the robot goes to these previously taught corners. Follow the steps for redefinition as if the station *was* mounted on its PyPlate (***), but do not place any hardware in the robot's path until the robot has moved to the first corner.
To avoid this completely, you may delete the station definition and define a "new" one.Please follow the following example :

Obs : (* *) : rack is the general term for defining either a single-position rack (i.e a station) or a 2 dimensional rack (rack for containers) .
(* * *) : a PyPlate ©) is the flat circular surface on which the robot is mounted and where the different stations are screw on .

1. Find the "Rack name" and "Index name" in PYSECTION.INFO

Examples :	rack name	index name
	R1:RACK.1	RACK.1.INDEX
	B:OVER.BALANCE	YES (i.e 1)

2. Select the **System** option, then make the **View Dictionary** selection .
3. Move the highlighted bar over **R1:RACK.1** (example) and press Ctrl+D to delete the rack .
4. If one is not already attached, run a Pysection command to get a hand (see section 2.4) .
5. Enter the robot teaching screen and use the function keys to bring the robot to a safe position, clear the Pysection involved .
6. Press **9** to define the rack .
7. At the blinking cursor, type the rack name as given in PYSECTION.INFO (e.g R1:RACK.1) and press **E**nter .
8. At the blinking cursor, type the rack index name given in PYSECTION.INFO (e.g R1:RACK.1) and press **E**nter .
9. Press **E**nter to verify that the proper hand is attached .
10. Follow the rack definition instructions keeping in mind that, in PyTechnology, racks are taught at a height where the bottom of the fingers are 1 cm above an uncapped container in that position .
11. Optional : after you have defined the rack, verify some positions from the teaching scrren by pressing **6** then entering the rack name, and an index value when prompted. Observe the robotic alignment over this position .

Clear region :

The clear region is the area surrounding the robot intended to be clear of harware. If station placement violates this clear region, you must be aware of the consequences. If placement in the clear region is necessary, it is best to strategically choose the first or last sector on the robot locator plate for positioning since the robot rarely travels through this area (beware of the chance of the "elbow" hitting objects when servicing stations across from the violated sector). You must add programming to keep the robot clear from obstructions .

3. THE CONTROLLER

3.1 Introduction

The whole system is operated through its controller.Operator interaction with the controller is made possible by having a PC connected to the controller .

A controller is a computer used to command the robot to position and orient the tool (the hand and the pippet in this system) or part correctly in space at all times. The controller functions as the "brain" of the robot. Robots today have controllers that are run by programs - sets of instructions written in code.

Almost all robots of today are entirely pre-programmed by people; they can do only what they are programmed to do at the time, and nothing else. In the future, controllers,with artificial intelligence (or AI) may allow robots to be more self-reliant and independent.

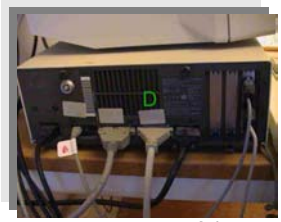
Pict. 8 : front view of the controller

3.2 Specifications

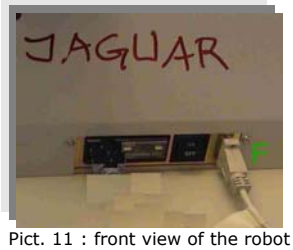
Dictionary space :	180 Kbytes per Controller Expansion Unit to a maximum of 540 Kbytes
Disk drive :	3 ½ floppy
Processor :	80C188 processor
Clock speed :	10 Mhz
Power :	100 / 120 volts 220 / 240 volts AC 50 - 60 Hz

3.3 Wire connections

Pict. 9 : rear part of the controller



Pict. 10: rear part of the PC



Pict. 11 : front view of the robot

A : socket (plug) connected to electric power
B : connection for the hand controller
C : to the 2nd power and event controller (björn)
D : to the PC (labeled there as D)
E : hävsorn (?. Not connected)
F : to the robot (see picture on left)
G : igelkott (? . Connected nowhere)
H : no card inside
I : to the vortex station
J : to the balance
K : to the Z410 capping station
L : to the power and event controller (fluga)
M : to the centrifuge station

4. STATIONS

4.1 Introduction

The combination of the different stations that surround the robot in the plate allows it to perform nearly any type of activity for pharmaceutical and chemical purposes, such as weighing fluids or solid substances, pippeting (*), mixing , evaporating solvents in the samples or capping, for instance .

(*) Pippeting : technical term used in chemistry for designating the act of sucking up a fluid with a tip from a container and then delivering it into another container .

Each one of these stations are located in a specific place on the plate. The plate is a round platform under the robot base that provides locating sectors for the physical connection of the stations (a sector is a numbered slot in the locator place). There are 48 sectors around the robot. Most stations consume from 2 to 6 sectors .

This chapter deals with all these different stations and provides detailed information on each one of them in order to have complete knowledge and control, and therefore being able to use them to perform all the activities as desired .

4.2 Original position of each station in the plate :

The following sectors show where the different stations must be placed around the robot :

Sectors :

From 48 to 1 (2 sectors) : LC Injection station (not used in this project)
2 to 3 : Disposal station
4 to 7 : Rack 1 (11 mm vial)
8 to 11 : Evaporation Station
12 to 14 : Dispenser (16 x100 mm vial). Not used in this project
15 to 16 : Liquid / Liquid Extraction
17 to 20 : Crimp Capping
21 & 22 : empty
23 to 25 : Dilute and Dissolve (Vortex Station)
26 to 29 : Weighing Liquid Transfer (Balance)
30 to 31 : Hand K - Pippeting (2 x 2 mm)
32 to 33 : Capping (16 x100 mm tube)
34 to 35 : Istd.hand
36 to 39 : Rack 2 (13 x 100 mm)
40 to 45 : Tumble Mixer (16 x 100 mm)
46 to 47 : Hand A

These stations can be removed from their original position, but then all the positions involving that station must be retaught (please see chapter 2.5 for further information) so that the system knows that a new station has been placed there .

4.3 Detailed information on each station :

HAND A :

Description :

The general purpose hand station provides the capability to grasp and move containers from 9 to 16 mm from station to station around the benchtop and is also used with the Capping Station .

Location : from sector 46 to sector 47 on the plate .

- Moved afterwards to sector 21-22 on the Pyplate (this operation has been performed in order to show how a station position must be retaught when is placed into another different location). See chapter 6.8 for more information .

Pict. 12 : hand attachment

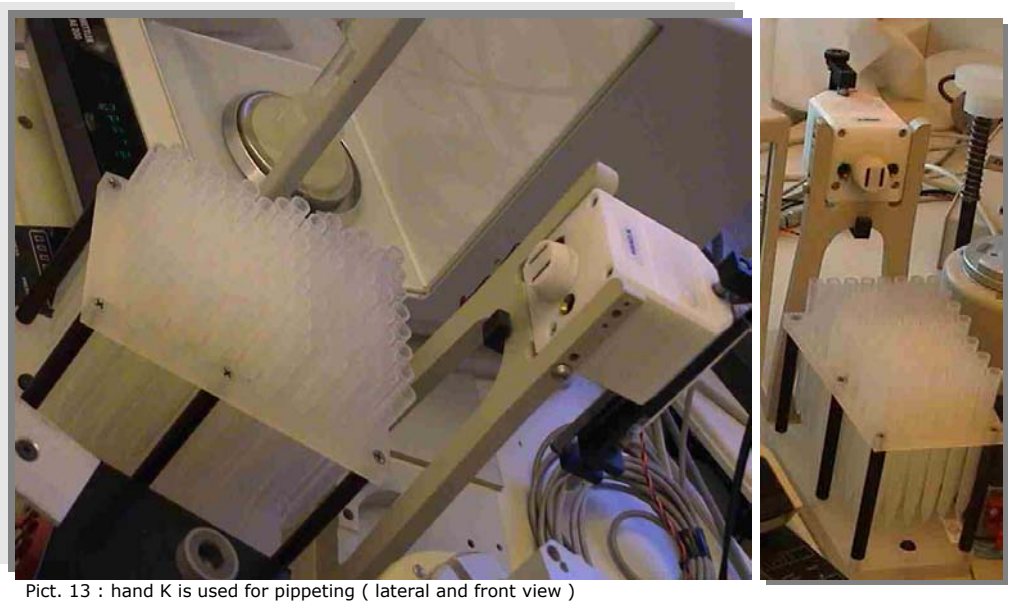
Instructions for this station :

get.hand.a :	the system checks to see if a hand is on the robot. If there is a hand and it is not the requested hand, this hand is parked. Afterwards the robot attaches the hand .
park.hand :	the system determines what hand is attached to the robot and performs the manipulations to park the hand. Then variable S:HAND.ID = 0 (no hand attached) .

HAND K (pippeting) :

Description : this station can transfer 0.2 to 2.0 ml of liquid using 2 ml disposable pipet tips between user-designated sample sources and destinations. The hand consists of a 2.5 ml syringe, automatic tip ejector, parking stationand a rack to hold 105 pipet tips .

Location : from sector 30 to sector 31 on the plate .



Pict. 13 : hand K is used for pippeting (lateral and front view)

Instructions for this station :

get.hand.k : the system checks to see if a hand is on the robot. If there is a hand and it is not the requested hand, this hand is parked. Afterwards the robot attaches the hand .

get.2ml.tip : the system determines whether hand K is on the robot.The system determines robot positioning and decides whether it needs to move to the appropiate tip rack clear region. The next tip is taken .

aspirate.2ml.tip : a *move.over…* command must be executed before this command, in order to place the robot over the sample source . The tip is moved into the liquid and a small air gap is drawn . The amount to be aspired is determined by the variable PIPET.VOLUME .

dispense.2ml.tip : a *move.over…* command must be executed before this command. The tip is moved to a height 1 cm above the final liquid level in the container below. The fluid is dispensed into the container. The amount dispensed is determined by the variable PIPET.VOLUME .

park.hand : the system determines what hand is attached to the robot and performas the manipulations to park the hand. Then variable S:HAND.ID = 0 (no hand attached) .

Variables for this station :

pipet.volume = 0.2 to 2.0 . This variable is used to specify the volume of liquid to be aspirated from the container below before executing the ASPIRATE.2ML.TIP command .

air.gap.1.volume.2ml = 0 to x. The defaul value for this variable is 0.025 .

aspirate.2ml.pause.time = 0 to x . The default value is zero. This parameter's value indicates how long (in seconds) the system will pause while aspirating liquid into the pipet tip .

dispense.2ml.pause.time = 0 to x . The default value is zero .

min.2ml.pipet.volume = 0.2 . The default value is 0.2 .

max.2ml.pipet.volume = 2.0 . The default value is 2.0 .

Demonstration program for the **PIPETTING 2 ml TIP station (hand K) :**

Name of the program : JOOCHAN.HAND.K
Description : this demostration program gets the pippeting hand (Hand K) and a 2 ml tip from the pippeting station. Afterwards, the tip gets stuck into the hand and a certain volume (decided by the user) is sucked from a test tube and released into another tube (both decided by the user as well) .

```
display.off
get.hand.k                          - takes the proper hand

get.2ml.tip                         - takes a tip from the station storage
prompt Insert source tube position ( Rack 2 ) : 1 to 50
input   rack.2.index                - links the number typed on the keyboard to
                                      the variable rack.2.index
move.over.rack.2                    - performs the movement
prompt Insert aspiration volume  : 0.2 to 2.0
input pipet.volume                  - links the number typed on the keyboard to
                                      the variable pipet.volume
aspirate.2ml.tip                    - sucks the volume ( saved in  pipet.volume ) from the
                                      test tube set by the variable  rack.2.index
prompt Insert destination tube position ( Rack 2 ) : 1 to 50
input    rack.2.index
dispense.2ml.tip                    - the volume sucked before is now put into another test
                                      tube ( set by thevariable rack.2.index )
dispose.to.waste                    - the dirty tip is taken to the trash store and dropped
park.hand
```

Observation : the instruction move.over.rack.2 has to be compulsory executed before aspirate.2ml.tip .

Warning : pipet.volume is set to 0 after executing the program. Therefore,if another application needs to handle this variable afterwards, it will have to be set again to a value greater than 0 .

Remember : the tip rack must be manually filled before starting an application .

ISTD.HAND :

Description : this Pysection (station) can transfer 0.010 to 0.200 ml of liquid using a syringe between the internal standard source and the sample. This station includes the syringe, a reusable luer-lock needle and a covered reservoir for the internal standard .

Location : from sector 34 to sector 35 on the plate .

Pict. 14 : front and lateral view of the istd.hand, used for liquid aspiration

Instructions for this station :

get.istd.hand : the system checks to see if a hand is on the robot. If there is a hand and it is not the requested hand, this hand is parked. Afterwards the robot attaches the hand .

park.hand : the system determines what hand is attached to the robot and performs the manipulations to park the hand. Then variable S:HAND.ID = 0 (no hand attached) .

aspirate.istd : a small air gap is drawn into the syringe. The cannula is moved into the liquid and the specified volume (variable ISTD.VOL) of liquid is aspirated into the syringe . The hand is moved over the reservoir and a second air gap is drawn. The value of ISTD.VOL is reset to zero .

dispense.istd : a MOVE.OVER… command (see chapter 2.4 "Commands and variables summary for the robot") must be executed before this one, as there are several places to deliver its contents, such as in a test tube placed in a rack or in a test tube placed in the vortex station, for example .
The cannula end is placed against the inside wall of the container below and the entire syringe contents are dispensed into the container. Finally, the cannula end is moved above the container .

Variables for this station :

istd.vol = 0.01 to 0.024 . This variable is used to specify the volume of internal standard to be aspirated from the reservoir below the hand. The value is st by using a math assignment, such as ISTD.VOL = 0.1.

istd.aspirate.pause.time = 0 to X (0 as the default value). This parameter indicates how long (in seconds) the system will pause while aspirating liquid into the syringe .

istd.dispense.pause.time = 0 to X (0 as the default value). This parameter indicates how long (in seconds) the system will pause while dispensing liquid into the syringe .

min.istd.vol = 0.01 (default value). It is the minimum value to be transferred by using this hand .

max.istd.vol = 0.024 (default value). It is the maximum value to be transferred by using this hand .

Demonstration program for the **INTERNAL STANDARD hand (aspiration):**

Name of the program : JOOCHAN.HAND.ISTD
Description : this demostration program gets the internal standard hand, asks for the volume to be aspirated and then delivers its contents into a tube .

```
display off
get.istd.hand
prompt Please insert the volume to be aspirated ( 0.01 to 0.2 )
input istd.vol                    - the amount to be aspirated is determined by this
                                    variable ( input by user in this case )
aspirate.istd                     - the volume set before is aspirated
prompt Insert destination test tube position ( Rack 2 ) : 1 to 50
input rack.2.index
move.over.rack.2
dispense.istd
park.hand
```

Observation : *rack.2.index* must be set before executing *move.over.rack.2*, since this movement instruction needs to know the tube position to deliver its contents . If not, the hand delivers the fluid in the position pointed by the previous value of *rack.2.index*, and a liquid spilling might be inminent .

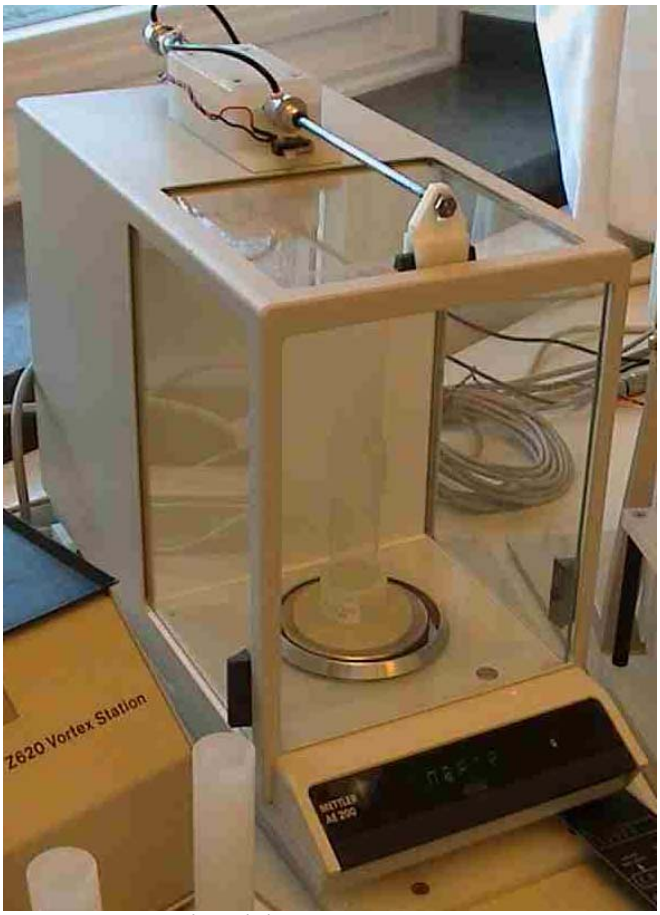
Warning : *istd.vol* is set to 0 at the end of the *aspirate.istd* command. Therefore,if another application needs to handle this variable afterwards, it will have to be set again to a value greater than 0 .

Remember : internal standard reservoir must be filled to the 50 ml mark before each run, and the maximum volume to be aspirated is 0.2 ml .

BALANCE :

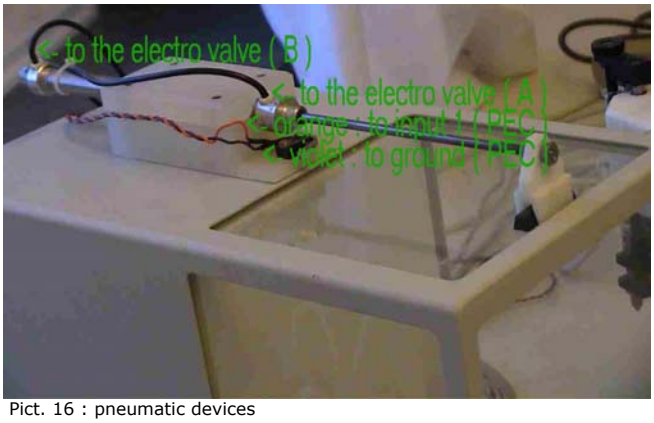
Description : this station is used for the analytical weighing of liquid samples and sample containers . A pneumatic balance door is included and allows sample containers to be top-loaded into the balance, preventing drafts from affecting the balance. Transfer of liquid samples is done by pippeting .

Location : from sector 26 to sector 29 on the plate .

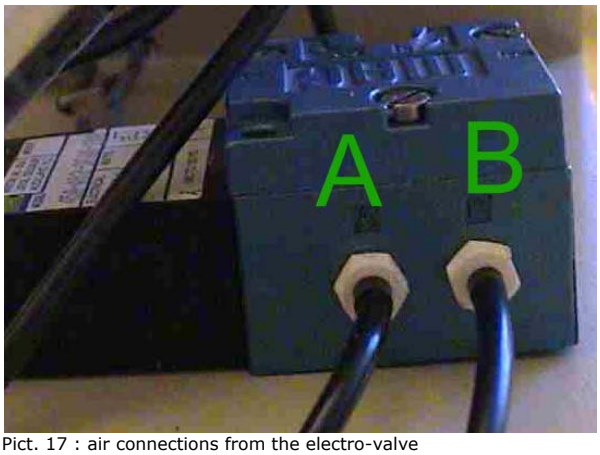


Pict. 15 : pneumatic driven balance

Connections :



Pict. 16 : pneumatic devices



Pict. 17 : air connections from the electro-valve

Pict. 18 : electric connections in the balance

↗ Observation : the balance must be manually set ! When the electric power is on for the first time the balance is off. Therefore the operator must push the black strip on the border of the device (just below the light panel) to switch it on .

Instructions for this station :

put.into.balance : the robot places the current tube in the hand into the balance .The pneumatic door is opened and the tube is put into. Then the door is closed .When the balance has become stabilized weight is taken .

obtain.weight : the weight value of the container is stored in a read-only variable called WEIGHT.VALUE. To get the weight shown on the screen command *printc* is needed .

move.over.balance : the robot moves the current hand (A, K or ISTD.HAND) over the balance and the balance door is opened . This is a useful instruction when delivering a pippeted sample into the balance tube, for example .

get.from.balance : the robot removes the container from the balance .

Variable for this station :

weight.value : the weight value is kept in this read-only variable .

Demonstration program for the **WEIGH station** :

Name of the program : JOOCHAN.BALANCE
Description : this program gets a test tube and displays its weight on the screen .

display off - it is used to make the current program lines not to appear on the screen

get.hand.a - gets the proper hand to take a tube afterwards

prompt Insert source test tube position (Rack1) : 1 to 50
- *prompt* shows a message on the screen

input rack.1.index - links the number typed by the user to this variable

get.from.rack.1 - takes a test tube from the position pointed by "rack.1.index"

put.into.balance - opens the door of the balance and the robot leaves the test tube inside in order to get the weight

obtain.weight - function for obtaining the weight of the sample
The value is saved to the variable "weight.value"

prompt The weight of the tube is :

printc WEIGHT.VALUE - *printc* shows on the screen the value of an internal variable, such as " weight.value "

get.from.balance - opens the door and the robot takes the tube

prompt Insert destination test tube position (Rack 1) : 1 to 50

input rack.1.index - the destination number typed is linked to this ariable

put.into.rack.1 - the tube is left in the position pointed by rack.1.index

park.hand - places the current hand into its original position (hand storage position) .

Observation : the weight is not taken unless the balance is stable . *put.into.balance* and *get.from.balance* are high-level instructions which have different sub-routines inside (error check for air pressure , open.door, close door …)

Warning : the proper test tube (glass tube) must be taken and from the proper rack (Rack 1) . If not there will be a crash when placing it into the balance .

Important : Make sure that the variable *air.confirm.sensor* is set to 0 !! Otherwise, error recovery subroutines will be performed .

EVAPORATION STATION :

Description : the Evaporation Pysection (station) fror 16 x 100 mm test tubes provides for the evaporation of solvent from containers placed in the heating block.
The temperature-controlled heating block is settable between ambient and 100 °C, and the temperature is set by the user. Solvent evaporation is accelerated by gas purge tubes which are lowered into the containers placed in the heating block. The user-supplied purge gas is used to pneumatically operate the station as well as dry the samples. Gas is only turned on when there are tubes in the station .

Location : from sector 8 to sector 11 on the plate .

Pict. 19 : front & top views of the evaporation station

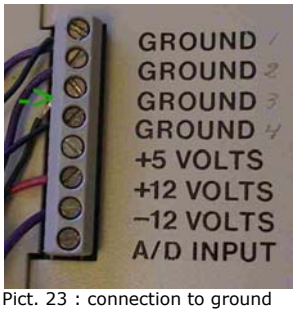
Connections :

Pict. 21 : PEC pneumatic connection

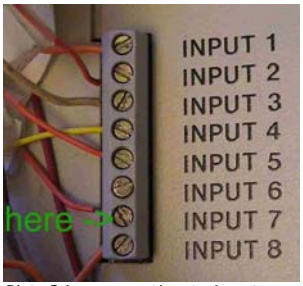
The black tube pointed by the arrow in picture 21 must go to the air input of the electro-valve in picture 20 .

Pict. 20 : connections to other devices

Pict. 22 : connection to the electric transformator



Pict. 23 : connection to ground



Pict. 24 : connection to input

Pict. 25 : connection to switches

Wire connection for the evaporation station

Black wire : ground
Orange wire : input 7
Brown wire : switch 5
Red wire : switch 6
(please look at the pictures above)

Instructions for this station :

put.into.evaporator : the robot performs the manipulations to put the current container attached into the station (in the position pointed by the variable EVAPORATOR.INDEX) . The evaporator arm is swung to the side, allowing access to the heater block . After this, the gas purge tubes are lowered and the gas is turned on .

↙ **Observation** : the evaporator heater must be manually turned on !

get.from.evaporator : the evaporator arm is swung pneumatically up and to a side and the tube is taken from the position pointed by EVAPORATOR.INDEX .

Variable for this station :

evaporator.index = 1 to 6 . This variable is used to specify the particular container position in the heater block. This value is set or modified by using a math assignment statement such as : EVAPORATOR.INDEX = 4 .

Demonstration program for the **EVAPORATION station :**

Name of the program : JOOCHAN.EVAPORATOR
Description : this program gets a cristal tube from the Tumble Mixer rack and places it into the Evaporator station in order to evaporate its contents. After 5 seconds it is taken back to the rack .

```
display off
get.hand.a
get.from.centrifuge
prompt Insert evaporatior position : 1 to 6
input evaporator.index          - links the number typed by the user to this variable
put.into.evaporator             - placed in the evaporator and the mechanism is activated
pause5                          - waiting for 5 seconds

get.from.evaporator
prompt Insert destination position ( Rack 1 ) : 1 to 50

input.rack.1.index
put.into.rack.1
park.hand
```

The previous high-level functions (*put.into* evaporator and *get.from.evaporator* include some basic instructions, such as :

e:mech.in	- the mechanism above rotates and the platform goes down. The metal tubes get placed into the test tubes and some air is blown into them to evaporate its contents .
e:mech.out	- the platform goes up and the rotatory mechanism turns .

Observation : a glass or plastic tube must be taken exclusively from the evaporator rack .Otherwise, a possible crash might be imminent .

CAPPING STATION (16x100 mm tubes)

Description : this capping station provides automated capping and uncapping of screw-capped 16 x100 mm test tubes. The success of capping and uncapping operations is internally verified.

Location : from sector 32 to sector 33 on the plate .

Pict. 26: grips and cap container

Connections :

This station is connected to the Z410 capping station (named " cykel "), and this last one is connected also to the controller (please see chapter *3.3 Wire connections* for further information).

Instructions for this station :

cap : the robot places the container to be capped into the capping station, takes the cap from the cap store and caps the container by moving yaws in a rotation movement .

uncap : the robot places the container to be uncapped into the capping station, takes the cap out from the tube and places it onto the cap store .

put.into.capper : the robot places the container in its hands into the capper and verifies that the jaws successfully close around the container .
This command is included in CAP and UNCAP command, so it is not necessary to write it to perform a capping .

get.from.capper : once the test tube has been capped or uncapped the robot moves down, takes the test tube and moves upwards. The CAP and UNCAP commnads do not remove the container from the capper. This action must be performed separately .

move.over.capper : this command is used when needing, for example, delivering the contents of the pippet tip into the tube (stuck in the jaws) .

Variables for this station :

cap.index =1 or 2 (default value = 1).This variable is used to specify which cap holder will be used for storing a cap .

cap.torque =1 to 3 (default value = 2). This parameter is used to specify the torque value to which tubes will be capped .

discard.cap.index = yes or no (default value = no). This parameter is used to specify whether the cap is discarded after it has been removed from the container .

Demonstration program for the **CAPPING (16x100mm Tube) station** :

Name of the program : JOOCHAN.CAPPER
Description : this program takes a test tube,cappes and uncappes it, and finally the tube is taken back to the rack (to the position pointed by the user) .

```
display off
get.hand.a
get.from.centrifuge      - takes a test tube from the Tumble Mixer station
put.into.capper
cap                      - performs capping

get.from.capper
put.into.capper
uncap                    - performs uncapping

get.from.capper
prompt Enter destination position ( Rack 2 ) : 1 to 50
input.rack.2.index
put.into.rack.2
park.hand
```

The previous high-level functions include some basic instructions, such as :

```
c: get.cap          - takes a cap from the storage place
c: park.cap         - brings the cap to the store place again
c: open.capper      - opens the claws of the capper to keep the tube
c: close.capper     - once the tube is released the claws get closed
```

Observation : in order to perform the capping, a plastic tube must be used and a plastic cap must be placed onto the station store .
This plastic tube must be located in the Tumble Mixer station, not in an ordinary rack. That is why the instruction *get.from.centrifuge* is used .

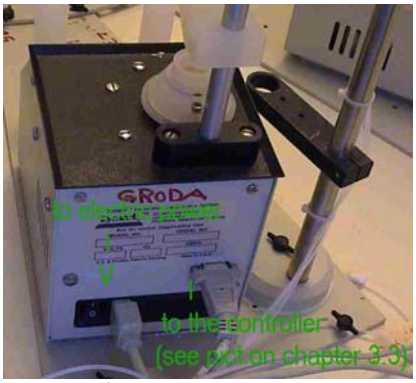
VORTEX STATION :

Description : this Dilute & Dissolve station is used to dispense up to 3 discrete liquids into a container. Liquid addition can be followed by a vigorous vortexing action (shaking) to spped mixing and dissolving. Liquid addition without vortexing and vortexing without liquid addition is also possible .

Location : from sector 23 to sector 25 on the plate .

Pict. 27 : vortex station and Input&Output tube containers

Connections :



Pict. 28 : electric connections

Instructions for this station :

put.into.vortex : the robot places a container into the vortex station .
get.from.vortex : the robot takes the test tube from this station. This command includes *vortex.off* .So, if get.from.vortex is executed when it is on, it will automatically stop .
vortex.on : this command makes the test tube be shaked. This activity is on until the user writes *vortex.off* or *get.from.vortex* .
vortex.off : if the votex is on, then it will stop .
move.over.vortex : the robot places the test tube just above the station.
vortex.timed.run : variable *vortex.time* must be linked to a value before executing this command. For example : vortex.timed.run = 4 .Then the test tube will be vortexed for 4 seconds .

v:on : low-level instruction (included in vortex.on) to start vortexing .
v:off : low-level instruction (included in vortex.off) to stop vortexing .

Variables for this station :

vortex.speed.1 = 40 (deful value)
vortex.speed.1.time = 1
vortex.time = x . This shows the time during which the vortex is on .

Demonstration program for the **VORTEX station :**

Name of the program : JOOCHAN.VORTEX
Description : this program takes a test tube from the position pointed by the user, places it into the vortex station and it is shaked during specified time. It is then taken and put back in the rack .

```
get.hand.a
prompt Enter source position ( Rack 2 ) : 1 to 50
input.rack.2.index
get.from.rack.2
put.into.vortex              - inserts the test tube into the agitator
prompt Insert duration time

input vortex.time            - links the typed number to this variable
vortex.timed.run             - enables the agitator to be on during the time
                               specified in the variable vortex.time
get.from.vortex              - gets the test tube from the agitator

prompt Enter destination position ( Rack 2 ) : 1 to 50
input.rack.2.index
put.into.rack.2
park.hand
```

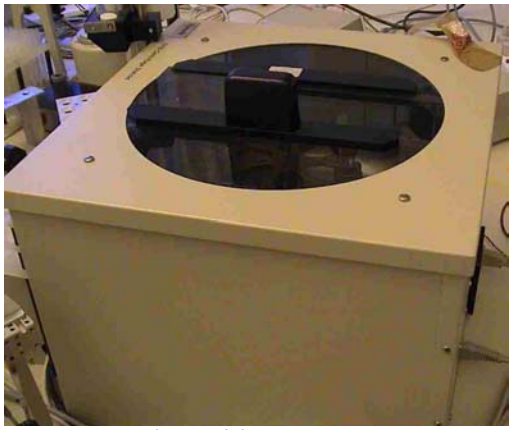
Observation : if there is already a test tube in the station and the function *put.into.vortex* is called, the robot will leave the tube into the *Input* container .Then it will take the current tube out of the station and will place it into the *Output* container.
After this, the robot will take the tube in the *Input* container and will place it into the station .

Warning : a crystal test tube must be put into Rack 2 before performing this program. If the tube is taken from Rack 1 the hand will crash against the station !

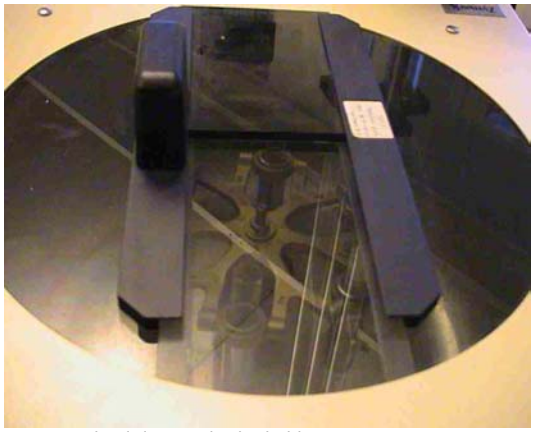
CENTRIFUGE STATION :

Description : this station provides automated centrifugation of samples. The electromecanically-operated door and rotor position indexing capability of the centrifuge allow interaction with this station .

Location : this station is NOT surrounding the robot on the plate. It is placed just behind the tumble mixer .

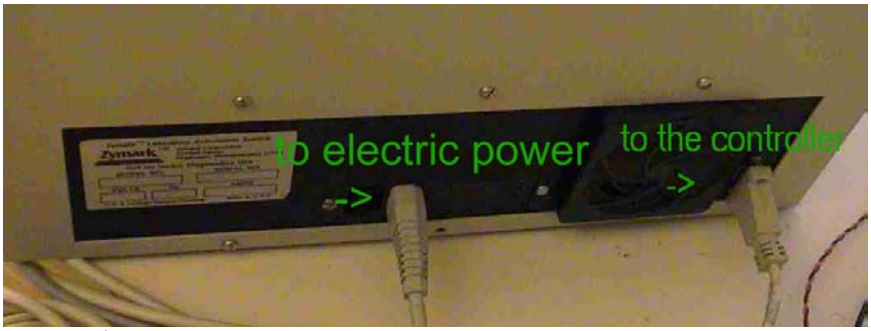


Pict. 29 : general view of the station



Pict. 30 : load door and tube holders

Connections :



Pict. 31 : electric connections

Instructions for this station :

put.into.centrifuge : the robot performs the manipulations to put the container into the input position (in the centrifuge rack) and afterwards places the test tube into the rotor position defined by the variable called *centrifuge.index* .

get.from.centrifuge : the robot takes the container from the position defined in the variable *centrifuge.index* and puts it into the output position in the centrifuge rack .

Variable for this station :

centrifuge.index = desired rotor position . This variable tells the system the place for the test tube to be put in .

Demonstration program for the **CENTRIFUGE station :**

Name of the program : JOOCHAN.CENTRIFUGE
Description : this program

```
display off
get.hand.a
prompt Insert source position (Rack 1): 1 to 50
input rack.1.index
get.from.rack.1
prompt Inster centrifuge position: 1 to 6
input centrifuge.index
put.into.centrifuge
pause10
f:vol.output = -1        –in the rack there is no container
get.from.centrifuge
prompt Inster destination position (Rack 1): 1 to 50
input rack.1.index
put.into.rack.1
park.hand
```

LIQUID / LIQUID STATION :

Description : this station performs the transfer of the desired extraction layer from an extraction container to a fluid-filled holding loop for subsequent dispensing into a collection container or waste.
You determine and program the depth at which the cannula goes into the extraction container,the volume to be extracted and the time required to extract the sample .

Location : from sector 15 to sector 16 on the plate.

✖ Observation : this station has not been used in this project, as a liquid feedback is needed.
An error message would appear if using this station without fluids .
Nevertheless, the instructions needed will be added if in a future this feedback is included .

Connections (PEC) :

black wire : to ground .
white wire : to input 3
red wire : to switch 2
orange : to input 2
brown wire : to switch 1

Instructions for this station :

aspirate.ll.cannula : the hand and container are brought under, then up to the cannula .
dispense.ll.cannula : the holding loop contents are dispensed into the collection container and any residual drop is touched off .

Variables for this station :

ll.cannula.depth = 0 to 15.0 (9.0 as the default value). This parameter is used to specify the depth in cm from the top of the container to where the end of the cannula will be placed for layer extraction .

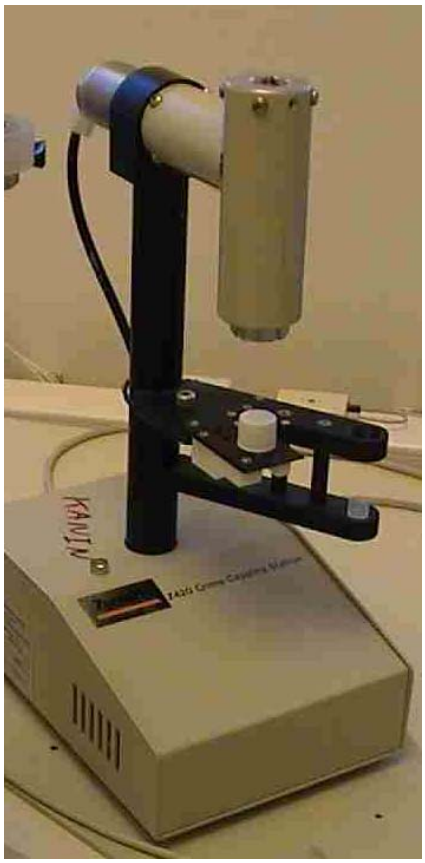
ll.cannula.volume = 0 to 9.8 (5.0 as the default value). This parameter is used to specify the volume of liquid in ml to be aspirated from the extraction container .

ll.aspirate.time = 1 to x (20.0 as the default value).

CRIMP CAPPING STATION :

Description : this station provides automatic filling and crimp capping of 11 mm vials . The Crimp Capping Pysection consists of a peumatically-driven crimp capper, vial cap holder and vial holder .
Cap presence in the holder is verified by a microswitch .

Location : from sector 17 to sector 20 on the plate .



Pict. 29 : crimp capping sation

✖ Observation : this station has not been used in this project, as a rack of 11 mm vials is needed in the system. Error messages appear when the system ask for them . Nevertheless, the instructions needed will be added if those vials are ever available.

Connections :

black wire : to switch 7
white wire : to switch 7 (?)
red wire : to input 6
green wire : to ground in the Power & Event Controller .

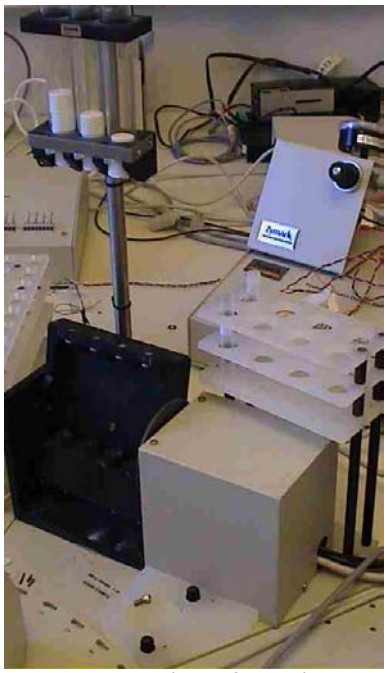
Instructions for this station :

get.from.crimp.capper : the robot performs manipulations to get the vial from the holder.
put.into.crimp.capper : the system determines if the cap has been removed, and if not, the robot readjusts the position of the vial and places the cap in the parking station. The vial is then placed in its holder .
move.over.crimp.capper : the robot positions the pipet tip into the tip guide .
crimp.vial : the vial is repositioned in the robot fingers and placed in the capper jaws. The vial is crimped .
uncap.vial : the robot removes and stores the cap in the cap holder .

TUMBLE MIXER (16X100 mm) :

Description : this station provides mixing of samples in 16 x 100 mm test tubes.The tumble mixer is typically used for liquid/liquid extraction procedures. Tests tubes are placed there and the turned end over end at 15 rpm (verification is performed to ensure that the tests tubes are capped before they are placed into the tumbling station) .

Location : from sector 40 to sector 45 on the plate .



Pict. 30 : mixer and centrifuge rack

✖ Observation : this station has not been used in this project, as polypropilene caps with polypropilene/foam liners (# Wheaton 242214) and 16x100 mm test tubes are needed to cap test tubes prior to mixing its contents . Nevertheless, the instructions needed will be added .

Connections :

black.wire : to ground (Power & Event Controller)
red wire : to switch 4
orange wire : to input 5
yellow wire : to input 4
brown wire : to switch 3

Instructions for this station :

put.into.tumble.mixer : the robot loads a container into the tumble mixer at the position defined by *mixer.index.* The robot moves into the station clear area and the system checks if there is an uncapped container in the hand.If so, the robot moves to the cap dispenser and executes the moves to cap the tube .

get.from.tumble.mixer : the robot gets the test tube from the tumble mixer (at the position defined by the *mixer.index* variable and removes the cap from the container just unloaded .

Variables for this station :

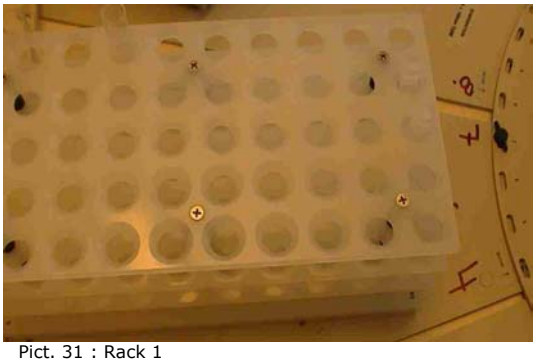
mixer.index = 1 to 4 . This variable is used to specify the position in the tumble mixer . This variable is set by using a math asignment. For instance : mixer.index = 3

RACK 1 (11 mm vial) :

Description : this section provides storage for 11 mm Gc vials. Each container in the rack (50 positions) has an associated volume and capped status that is updated when the container is taken and moved around the benchtop . The test tubes held in this rack must be taken with hand.a .

Location : from sector 4 to sector 7 on the plate .

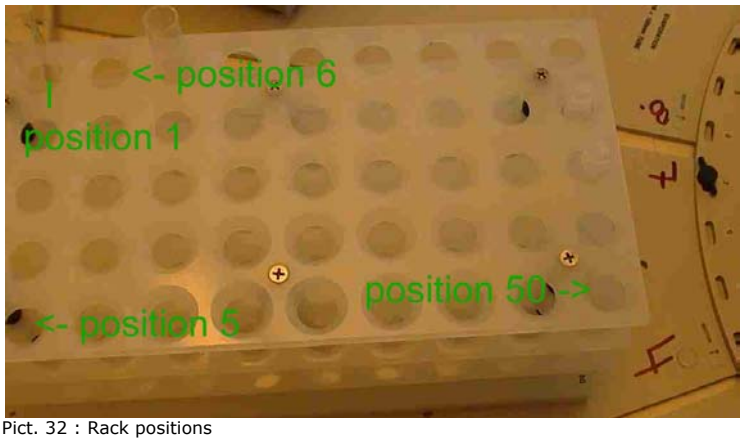
- Moved afterwards to sector 17-20 on the Pyplate (this operation has been performed in order to show how a station position must be retaught when is placed into another different location). See chapter 6.8 for more information .



Pict. 31 : Rack 1

Rack definition :

Positions in the rack are numbered from position 1 to position 50 like this :



Pict. 32 : Rack positions

Instructions for this station :

get.from.rack.1 : a container is taken by the hand from the rack at the position defined in the variable *rack.1.index* .

put.into.rack.1 : the robot puts the container into the rack at the position defined in the variable *rack.1.index* .

Variables for this station :

rack.1.index = from 1 to 50 . This is where the current test tube must be taken from or to be placed in .

initial.volume.rack.1 = 0 to 12 (ml) (0 as the default value).This is the initial starting volume in ml for each tube in the rack . If the rack is going to be empty of containers, this variable must be set to –1 .

container.capped.rack.1 = YES or NO (NO as the default value). If set to NO, it shows that the rack is full with uncapped containers .

Internal variable :

r1:vol =
= 0 : there is a container (but empty)
= positive number : there is a container (full)
= -1 : there is no container

This variable might be modified in those programs when an error appears and there is no quick solution for it. For instance, if the system says that there is no container at a position but ther is actually one, this variable might be modified in order to make the system know that .

Demonstration program for the **RACK 1** :

Name of the program : JOOCHAN.RACK.1
Description : this demostration program gets hand A and asks for the source position where the test tube is. The robot gets it and puts it into the same rack (in the position entered by the user) .

```
display off
get.hand.a
prompt Insert source position ( Rack 1 ) : 1 to 50
input rack.1.index
get.from.rack.1          - gets the test tube into the position pointed by
                           rack.1.index
prompt Insert destination position ( Rack 1 ) : 1 to 50
input rack.1.index
put.into.rack.1
park.hand
```

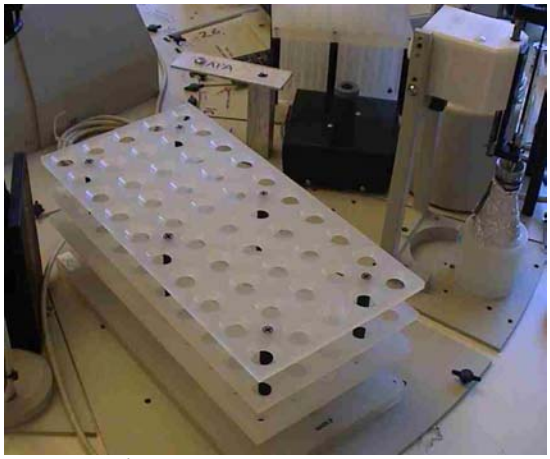
Observation : rack.1.index must be set before executing get.from.rack.1 order. If not, the test tube will be taken from the last position pointed by rack.1.index .

Remember : cristal tubes are kept in this rack . Do not place any other type of tube (plastic tubes), since the robot is not able to take them from here. Otherwise an error message will appear .

RACK 2 :

Description : this rack is used to keep 13 x 100 mm test tubes. Each container in the rack (50 positions) has an associated volume and capped status that is updated when the container is taken and moved around the benchtop . The test tubes held in this rack must be taken with hand.a .

Location : from sector 36 to sector 39 on the plate .



Pict. 33 : Rack 2

Instructions for this station :

get.from.rack.2 : a container is taken by the hand from the rack at the position defined in the variable *rack.1.index* .

put.into.rack.2 : the robot puts the container into the rack at the position defined in the variable *rack.1.index* .

Variables for this station :

rack.2.index = from 1 to 50 . This is where the current test tube must be taken from or to be placed in .

initial.volume.rack.2 = 0 to 12 (ml) (0 as the default value).This is the initial starting volume in ml for each tube in the rack . If the rack is going to be empty of containers, this variable must be set to –1 .

container.capped.rack.2 = YES or NO (NO as the default value). If set to NO, it shows that the rack is full with uncapped containers .

Demonstration program for the **RACK 2** :

Name of the program : JOOCHAN.RACK.2
Description : this demostration program gets hand A and asks for the source position where the test tube is. The robot gets it and puts it into the same rack (in the position entered by the user) .

```
display off
get.hand.a
prompt Insert source position ( Rack 2 ) :1 to 50
input rack.2.index
get.from.rack.2
prompt Insert destination position ( Rack 2 ) : 1 to 50
input rack.2.index
put.into.rack.2
park.hand
```

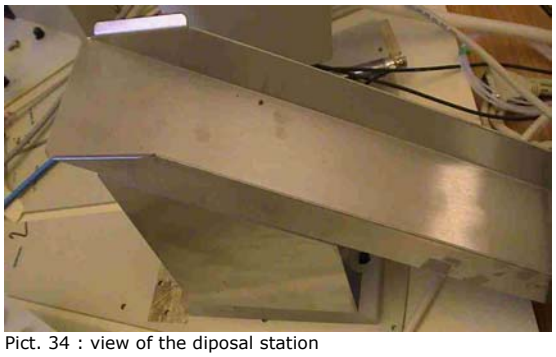
Observation : rack.2.index must be set before executing get.from.rack.2 order. If not, the test tube will be taken from the last position pointed by rack.2.index .

Remember : cristal tubes are kept in this rack . Do not place any other type of tube (plastic tubes), since the robot is not able to take them from here. Otherwise an error message will appear .

DISPOSAL :

Description : this section provides for the discarding of used disposables sucha as pipet tips, extraction columns, screw caps … A waste bag must be attached under this station to receive the discarde items .

Location : from sector 2 to sector 3 on the plate .



Pict. 34 : view of the diposal station

Instructions for this station :

dispose.to.waste. The robot deposits the item in the hand (pipet tip or test tube) in this station .

Demonstration program for the **DISPOSAL station :**

Name of the program : JOOCHAN.DISPOSAL
Description : this program gets a test tube from rack.1 and takes it to the waste container, where used tubes are kept .

```
display off
get.hand.a
prompt Insert  source position ( Rack 1 ) : 1 to 50
input.rack.1.index
get.from.rack.1
dispose.to.waste
park.hand
```

4.4 General demonstration program using all the available stations :

Demonstration program for the **WHOLE equipment :**

Name of the program : JOOCHAN.DEMO.1
Description : in this program all the available stations have been used and all the commands to make them work have been included.

```
display off
rack.1.index = 1
rack.2.index = 1
pipet.volume = 0.5
istd.vol = 0.1
evaporator.index = 1
centrifuge.index = 2


get.hand.a
get.from.rack.1
put.into.balance
obtain.weight
prompt Weight :
printc weight.value
joochan.before = weight.value
get.from.centrifuge
put.into.capper
uncap
park.hand
get.hand.k
get.2ml.tip
move.over.capper
aspirate.2ml.tip
move.over.balance
dispense.2ml.tip
dispose.to.waste
bd:close.door
park.hand
prompt New Weight :
```

Demonstration program for the **WHOLE equipment (cont) .**

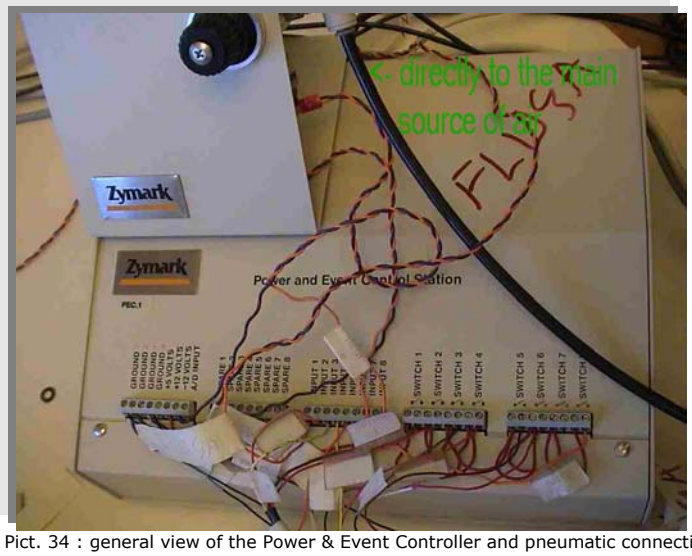
```
printc weight.value
joochan.after = weight.value
joochan.added = joochan.after – joochan.before
prompt Added liquid :
printc joochan.added
get.hand.a
move.over.capper
cap
get.from.capper
put.into.rack.2
get.from.balance
put.into.vortex
vortex.speed.1 = 50
vortex.on
park.hand
get.istd.hand
aspirate.istd
move.over.vortex
vortex.off
dispense.istd
vortex.speed.1 = 100
vortex.time = 10
vortex.timed.run
park.hand
get.hand.a
get.from.vortex
put.into.evaporator
pause10
get.from.evaporator
put.into.centrifuge
timer ( 1 ) = 20
wait.for.timer.1
get.from.centrifuge
put.into.rack.1
park.hand
```

4.5 GENERAL VIEW OF THE ROBOT AND THE STATIONS :

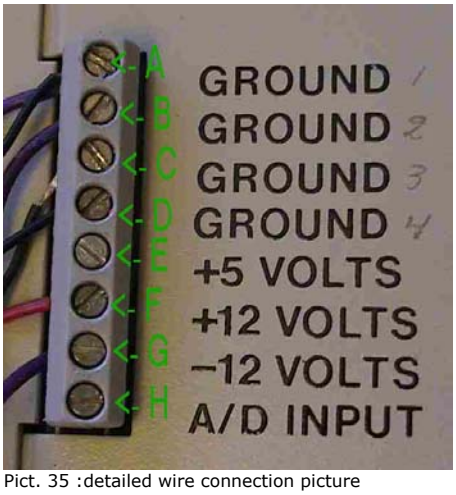
5.POWER & EVENT CONTROLLER (PEC) :

5.1 Introduction

The Power and Event Controller (PEC) interfaces with devices that supplement the operation of other stations. The Power and Event Controller provides control for devices used in Crimp Capping, Evaporation, etc..
Each Power and Event Controller provides programmable control of 8 output switches, 8 logic inputs, 2 AC On/Off power outlets, 1 variable AC power outlet and an analog/digital input.
A +5 v, +12 v and –12 v power supply is also included, and 8 Spare connectors are also available for use.
These are used as terminals when an internal option such as the Preamplifier Modulle is installed within thr PEC .
A Power and Event Controller's facilities may be used by different stations. The assignment of a PEC and its programmable connections take place when the sation requiring their usage is installed in the system.
The PEC facilities are operated by the individual sections.
An Air Monitor Manifold that provides a confirm sensor, filter, regulator and air line connections manifold is included with the Power and Event Controller .



Pict. 34 : general view of the Power & Event Controller and pneumatic connection

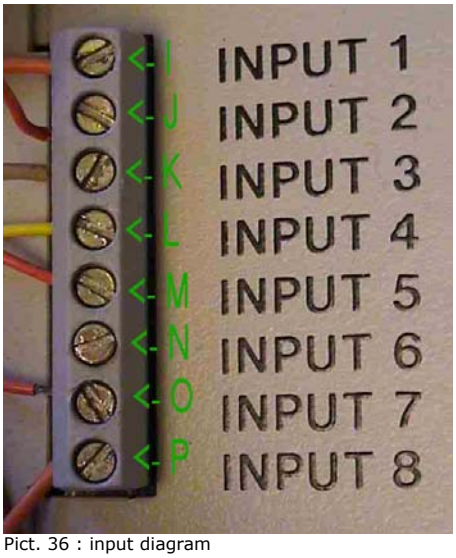


Pict. 35 :detailed wire connection picture

Connections :

A : black wire : from the liquid / liquid extraction station
violet wire : from the electro valve
B : violet wire : from the balance
C : black wire : from the evaporation station
D : black wire : from the tumble mixer
E : not connected
F : to switch 1.Power supply (*)
G : to the air sensor
H : not used

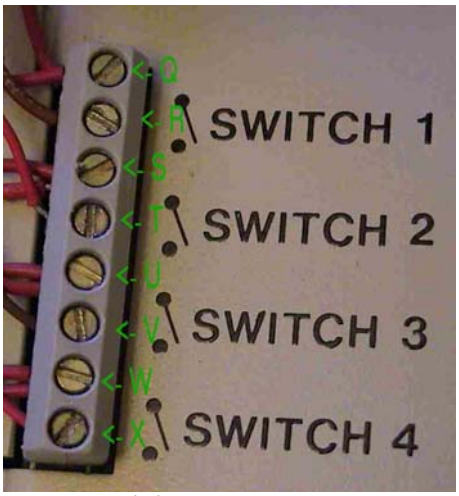
(*) There is a serial conection from the switch1 that takes +12 volts to the other switches (switch 2 to switch 8). Otherwise, all the switches should be connected to +12 volts !



Pict. 36 : input diagram

Connections :

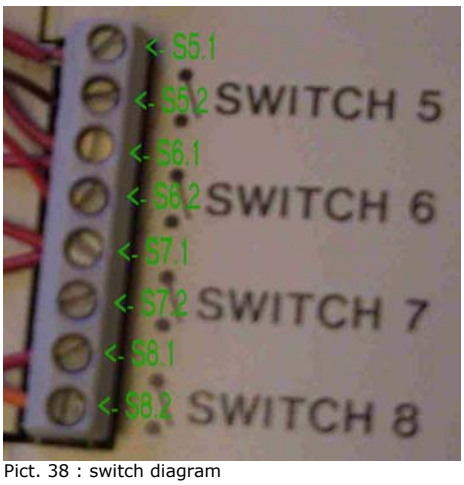
I : orange wire : from the balance
J : orange wire : from the liquid / liquid extraction station
K : white wire : from the liquid / liquid extraction station
L : yellow wire : from the tumble mixer station
M : orange wire : from the tumble mixer station
N : red wire : from the crimp capping station
O : orange wire : from the evaporation station
P : orange wire : from the electro valve



Pict. 37 : switch diagram

Connections :

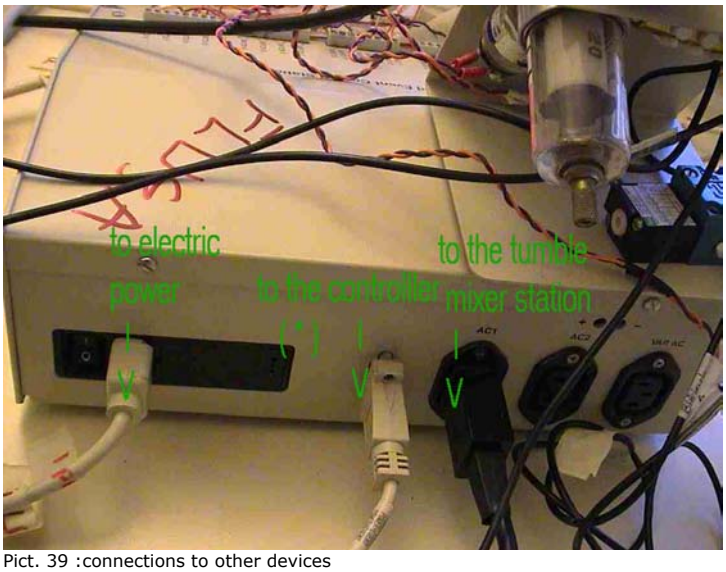
Q : connected to +12 volts
R : brown wire : connected to the liquid / liquid station
S : connected to Q (+12 volts)
T : red wire : connected to the liquid / liquid station
U : connected to S (+12 volts)
V : brown wire : connected to the tumble mixer station
W : connected to U (+12 volts)
X : red wire : connected to the tumble mixer station



Pict. 38 : switch diagram

Connections :

S5.1 : connected to W (+12 volts)
S5.2 : brown wire : connected to the evaporation station
S6.1 : connected to S.5.1 (+12 volts)
S6.2 : red wire : connected to the evaporation station
S7.1 : connected to S6.1 (+12 volts)
S7.2 : white wire : from the crimp capping station
S8.1 : connected to S7.1 (+12 volts)
S8.2 : orange wire : connected to the electro-valve power supply.



Pict. 39 :connections to other devices
(*) See chapter 3.3 -Wire connection (named "L")

6. SOFTWARE :

6.1 Introduction

This equipment combine robotics, computer and laboratory stations to automate laboratory procedures.

The robot transports the sample between stations, the computer controls hardware execution and sofware execution, and the laboratory stations perform basic operations, such as pipetting, mixing, diluting, centrifugation, extraction…

Before the system is running and performing any type of chemical / pharmaceutical operations, the system must load specific software (called EASYLAB), which allows the user to program the robot and stations as desired .
As shown in chapter 4, each station has its own pre-programmed commands (get.hand.a for instance), which are kept in a dictionary .
This dictionary has to be created when the system is running for the first time by placing in the PC all the disks delivered by Zymark and “Pyappending” their contents.

🔔 Observation : No program installing is needed !

> This is because this system had already been working before (in AstraZeneca).Therefore, all program installation had already been performed and dictionaries had been created. The final user has only to make his own programs according to his plans by using the instructions in the dictionary .

This section describes the system software and the procedure to make different programs to use the stations, as well as how to modify existing programs, how to save them or how to check the correct working process of a new program step by step .

6.2 Software terms :

EASYLAB PLUS : is the software provided allowing inmediate operation. Software features include : simple,standarized command structure; ability to test a station's function while configuring the benchtop; tracking a container's status (i.e volume) while it moves from station to station; error recovery and confirm verification techniques .

LOAD is a disk drive command which causes the contents of an EasyLab formatted disk to *replace* the entire contents of the controller's current dictionary .

CLEAR REGION is the area surrounding the robot, intended to be clear of harware . This assures clearance for the robot arm as it moves throughout ist 360° rotation .

APPEND is a disk drive command which causes the dictionary of a disk to be added to the Controller's current dictionary . Current dictionary entries with the same name as an incoming entry will be overwritten .

PYAPPEND is a disk drive command which causes the dictionary os a disk to be added to the Controller's current dictionary and also establishes the physical location of the Pysection in a user-specified sector around the robot locator plate . Current dictionary with the same name as an incoming entry will be overwritten (such as SETUP, TEST and PYSECTION.INFO) .

SETUP is a program included on each EasyLab disk (supplied by Zymark) that allows the user to manually align the Pysection (station) that was *last* PYAPPENDed with respect to the robot in the chosen sector .

TEST is a program included on each EasyLab disk that allows the user to test the operation of the station that was *last* Pyappended .

PYSECTION.INFO is a program included on each EasyLab disk that provides specific documentation for the station that was *last* Pyappended. This program is intended to be read on-screen or printed .

6.3 Zymate Files

There are several MS-DOS file types used by the zymate System V controller :

- files installed on the PC hard drive or on a zymate boot diskette when PCSETUP run (i.e SYS5.EXE, SYS5.HLP ...)
- files installed in controller memory when you boot the controller from the System Disk or a Controller boot diskette (ZYO.ZOS, INTP.ZMD...).

  These files MUST EXIST on all controller boot diskettes in order to successfully operate the system :

  ZYO.ZOS is a file that contains the zymate Operating System.
  INTP.ZMD interprets all Easy Lab commands and variables
  EXEC.ZMD provides screen information to the PC
  PCPRINT.ZMD gives access to the parallel printer port on the PC

- files containing a zymate dictionary (.ZYD file) : are created when a controller dictionary is saved to a diskette. Typical .ZYD files contain individual station dictionaries, configuration, application, or working dictionaries .

  ZYD files are not automatically loaded into memory when the controller is booted. Instead, they are loaded into the controller by using the **Load Dictionary** or appended using the **Pyappend Dictionary** selection from the System V On-line menu (see forward) .

6.4 On-line / Off-line menus :

On-line menu allows the user to interact with the controller and the entire System.
Programs can be executed in manual control, dictionaries can be loaded directly from the controller or a new program can be saved in the controller, for instance .

To achieve that, it is required that the controller is connected to the PC and be idle (not running), whereas in Off-line menu the PC is not connected to the controller, but allows the user to edit programs, view/change data or view dictionaries .

6.5 Dictionary :

The dictionary is the portion of memory in the controller that stores the information involving hardware and software information.
Hardware information is that one about the different stations (positions and variables) and software information deals with EasyLab commands and programs written or installed in the system .
Only one dictionary exists in the controller at a time, but different dictionaries can be saved as separate MS-DOS files to a single disk .
The operations allowed are : loading or clearing a dictionary from a diskette, modifying and saving it, or just viewing its contents .
Observation : when a new program is created it must be saved into the current dictionary before exiting the system and switching it off. If not, it is lost !

6.6 Getting started :

All the programming we have done is **On-line**. This chapter deals about how to succeed in programming and editing existing programs, but first it is necessary how to start .
First of all switch all the equipment on : the robot, the controller, the PC and the different stations to be used .
Then a screen like the following one will appear :

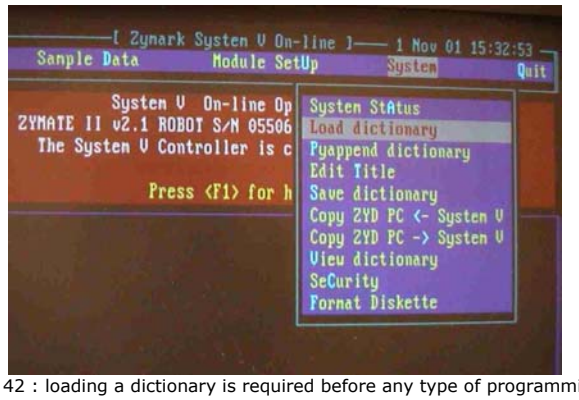
Pict. 40 : introduction screen

Press “ System V On-line ” option. The Main Menu will be displayed :

Pict. 41 : System V Online Main menu

If not, the next message is displayed : “ The System V Controller is not responding. Press a key to continue ” .
To solve this, insert a diskette that contains a dictionary in the controller, press the RELOAD button (on the front panel of the controller) and press any key. Then the previous screen will appear (picture 41) .

In order to perform any activity, loading an existing Dictionary is required before .
In the Main Menu go to " System " and press the " Load Dictionary " option.



Pict. 42 : loading a dictionary is required before any type of programming

A list of all the existing dictionaries will be displayed. Select a dictionary (Joochan.zyd in this case) and press Enter .

Pict. 43 : name of the dictionary saved in a diskette

Now all the variables and programs concerning the robot and the stations are available and loaded into the system.

Press Escape and come back to the Main Menu .

On the top bar of the Main Menu (pict.41) different programming sub-menus are shown :

Methods : 3 possibilities are given :
- Executing a Program
- Edit Programs
- Manual Control

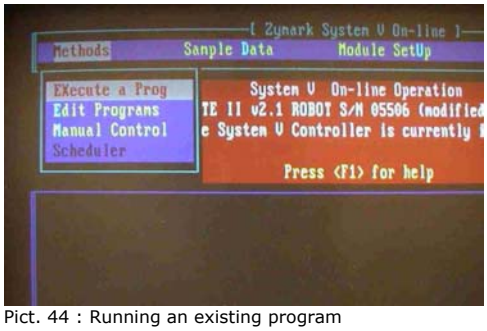
Sample Data : is were all the variables are kept .

Module Setup : all the information concerning the modules is there. This section is needed when a station must be placed somewhere else .

System : dictionary activities must begin at this point .

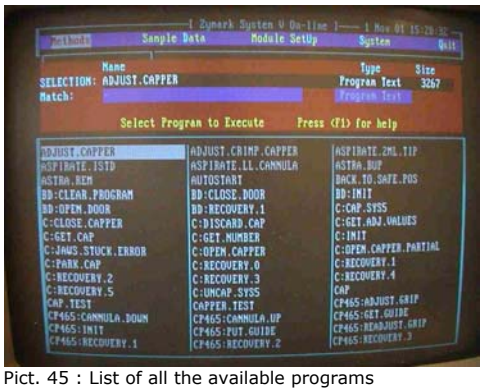
Quit

By entering in Methods and selecting Execute a Prog ,



Pict. 44 : Running an existing program

a list of all the different programs is shown



Pict. 45 : List of all the available programs

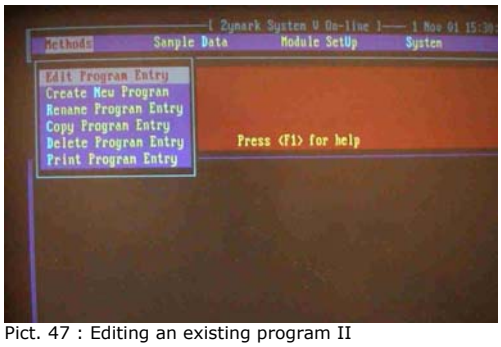
Place the lighted bar over the program you want to run and press enter (make sure before this that all the devices have the elements to work properly, such as pressured air or electric power).

By entering in Methods and selecting Edit Programs , the programmer is allowed to modify an existing program. This is useful when a program that has recently been created need some changes due to failure in working .

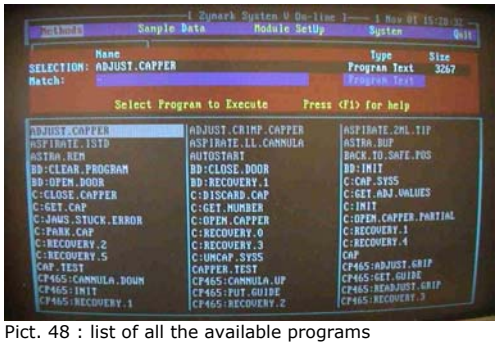
Pict. 46 : Editing an existing program I

By doing so, a new sub-menu is unfolded :

Edit program entry : a list with all the existing programs is shown and by pressing again on a selected name, the program can be modified (see pictures 47 and 48) .
Create New program : new programs are created here .
Rename Program entry : if the name of a program is to be changed .
Delete Program Entry : to delete an existing program .
Print Program Entry : to print a program .



Pict. 47 : Editing an existing program II



Pict. 48 : list of all the available programs

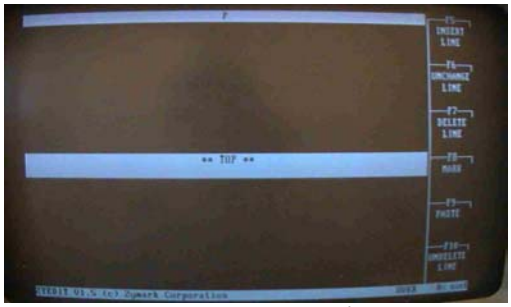
If the user wants to make his own programs, next step is necessary :

Pict. 49 : Creating a new program I

The system will ask for a name for the new program :

Pict. 50 : Creating a new program II

And afterwards, the user may start programming :



Pict. 51 : main programming menu and options

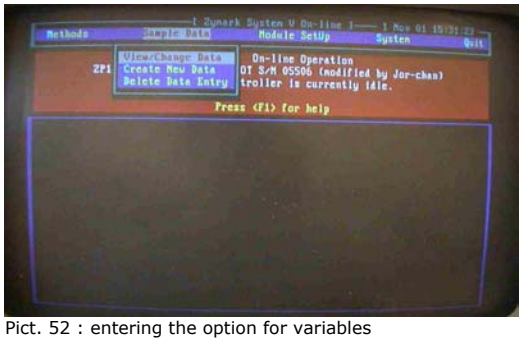
PROGRAMMING KEYS :

F5 : Insert Line
F6 : Unchange Line
F7 : Delete Line
F8 : Mark
F9 : Paste
F10 : Undelete Line

Variables :

Variables are kept in this section. They can be created or just read .For instance, if a mathematical variable is needed, it must be created in this menu .

Go to Sample Data in the Main Menu and press enter. The following screen will be shown :



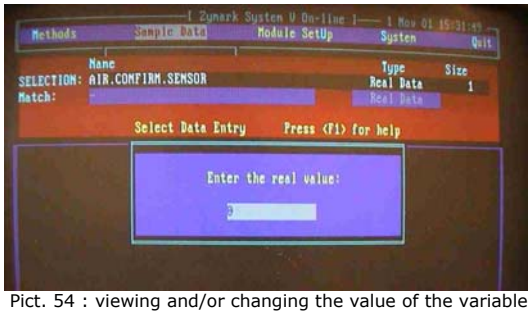
Pict. 52 : entering the option for variables

For example, if the variable to be changed is air.confir.sensor, place the highlighted bar on it and press Enter .

Pict. 53 : content of View/Change data menu

The value of this variable can be now read or changed .

Observation : this option is very useful for variables wich deal with sensors. For instance, if there is no pressured air for the stations, the system will display a message in which it is said that some air is needed. The solution is to Edit that program and see where the variable for that error appears. Then you must go to Sample Data menu and change the variable to the proper value so that the problem does not appear any more .



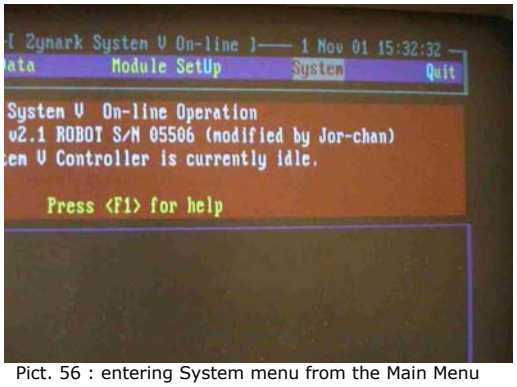
Pict. 54 : viewing and/or changing the value of the variable

Next section shows how a new variable is created. For instance, in the demonstration program for the whole equipment (chapter 4.4), variable joochan.added has been created .

Pict. 55: creating new data

System :

In this section dictionary information and related items are saved. After creating a new program, it must be saved into the dictionary. Otherwise, it is lost when switching the controller off .



Pict. 56 : entering System menu from the Main Menu

A complete section is unfolded :

The main functions are :

System Status : this section is only about the amount of memory and memory available .

Pict. 57 : introduction screen

Load Dictionary : enter this menu each time you run the system for the first time .

Pyappend Dictionary : when some information in another different dictionary is to be included in the current one, you must append its contents . It is also needed when re-programming a previously removed station (because initial installation disks must be appended to the current dictionary) .

Pict. 58 : appending a dictionary

Save Dictionary : information in the system is saved into the disk in the controller .

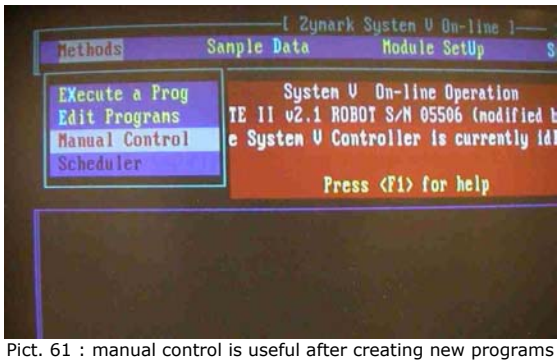
Pict. 59 : saving a dictionary

A name for the dictionary will requested. If writing again an existing name, the old dictionary will be overwritten) :

Pict.60 : enter the dictionary name

Copy ZYD PC <- System V : for copying information from the controller to the PC .
Copy ZYD PC -> System V : for copying information from the PC to the controller .
View dictionary : variables and programs are shown there.

Finally, the last important section to be shown is Manual Control . It allows the user to execute a program step by step .
The robot must be connected and instructions must be written on the screen one at a time .



Pict. 61 : manual control is useful after creating new programs

6.7 Loop & Conditional commands :

Loop and conditional commands are commonly used when programming.
They allow to repeat an action for a concrete number of times (loop sentences) or execute an action only when a specific condition is true (conditional sentences) .

In this chapter the main instructions are explained, and by moving forwards for some pages, demonstration programs are given .

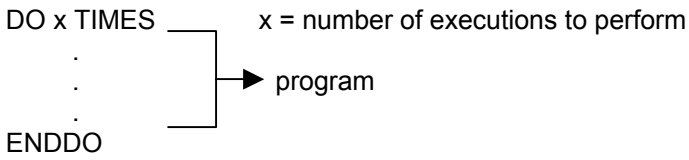
6.7.1 **Loop** commands :

DO / ENDDO :

Description:

" DO Loops" are used within a program to execute a group of statements a specific number of times.

Format:



Remarks:

The DO and ENDDO statements "frame" the section of program they affect.
All DO statements are not affected by an ENDDO statement within the same program. This is, they are program-level sensitive.
An active DO statement is not affected by an ENDDO statement appearing in another program.
If a number is calculated to specify the number of loops to execute, and that number is a non-integer value, that value is rounded to the closest integer.
Under DISPLAY OFF, the number of the loop being executed is still displayed.
DO Loops can be exited (using a conditional or GOTO statement) before the loop has completed.

GOTO :

Description:

This command is used in a EasyLab statement to branch unconditionally out of the normal program sequence to a specified line number.

GOTO statements are used primarily to alter the sequence of operation. This involves jumping to another location in the program to execute another program or module action, or, pausing system operation.

Format:

GOTO x x = constant or variable representing a line number

Remarks:

This command affects the sequence of execution of EasyLab program statements. A GOTO statement causes program execution to “jump” to the statement in that the program that is labelled with the specified line number. Program execution continues from that statement.
EasyLab Program lines are executed in the order they appear in the program. Although line numbers are not required to determine the operational flow of the program, they are used for jumping purposes. These numbers which can range from 1 to 65535 are used only as reference points for program branching.
GOTO statements are program-level sensitive. A GOTO statement must refer to a numbered statement whitin the same program.
The numbered statement referred to by a GOTO statement is always searched for from the beginning of that program.
GOTO statements should be programmed logically to avoid creating endless loops.
If the statement number is calculated using an EasyLab math expression and a non-integer value results, that calculated value is rounded to the nearest integer. The integer value must range between 1 and 65535.
A line number soecified by a GOTO statement mus be labeled a statement appearing within *that* program.
The same line number may be used in several different programs. Branching always occurs within the same program.

If the same line number appears multiple times in a single EasyLab program, the first occurrence of that number from the beginning of the program is the one referenced.
There are 5 spaces reserved at the beginning of each statement for line numbers.
A line number may appear before each statement in EasyLab program, but a program may be written without any line numbers.
A line number does NOT cause an EasyLab program to be excluded by sequential line numbers.
Line numbers are used as reference markers for program branching statements.

IF / THEN :

Description:

This command is used in an EasyLab statement to make a decision regarding program flow based on a result returned from an expression. The change in program flow can incolve jumping to another program line, executing another program or module action, performing a calculation, specifying a module action, or pausing system operation. Normally, EasyLab program lines are executed in the order they appear in the program. When the IF/THEN statement involves jumping ro another program line, a *line number* is used. Line numbers (which can range 1 to 65535) are used only as reference points for program branching by IF/THEN and GOTO statements.

Format:

IF *expression comparator expression* THEN *statement*

where *expression* is either a constant, a real data variable, or a module command variable (input or bi-directional only)
where *comparator* is either =, <>, >, < , <= or > =
where *statement* is either a line number, module command, program, math expression, or an EasyLab language command (except an IF/THEN statement or DO LOOP) .

Remarks:

If the condition is not satisfied, the statement following “THEN” is ignored. Program execution continues with the next statement in the EasyLab program.
IF/THEN statements are program-level sensitive when the statement following “THEN” refers to a numered statement. When “THEN” is followed by a number, the numbered statement must appear within the same program as the IF/THEN statement and be preceded with the line number specified.
If a line number is calculated following “THEN”, and a non-integer value results, that value is rounded to the nearest integer.
If a line number is calculated following “THEN”, the “THEN” mus be followed by “GOTO”.
If a line number is specified following “THEN”, “GOTO” is not necessary if a constant is used.
A line number specified by an IF/THEN statement must label a statement apearing within THAT program.
The same line number may be used in several different programs. Branching always occurs within the same program.
If the same line number appears multiple times in a single EasyLab program, the first occurrence of that number from the beginning of the program is the one referenced.
There are 5 spaces reserved at the beginning of each statement for line numbers.
A line number may appear before each statement in EasyLab program, but a program may be written without any line numbers.
A line number does NOT cause an EasyLab program to be excluded by sequential line numbers.
Line numbers are used as reference markers for program braching statements.

6.7.2 **Time** commands (pausing) :

TIMER (x) / WAIT FOR TIMER x :

Description:

There are eight internal timers available to the operator as EasyLab Language Commands that may be used to delay system operation or time events during the running of a program. Module may operate while timers are active. More than one timer may be in use at the same time.

Format:

To set timers:

TIMER(x) = *math expression*

where *x* is equal to 1 through 4 (1 through 8 in non-PyTechnology systems) and math expression is any constant or variable representing time in seconds.

To wait for timers:

WAIT FOR TIMER x

where *x* is equal to the timer set – 1 through 4 or 1 through 8 in non-PyTechnology systems. “x” should NOT be enclosed in parentheses in this format.

Remarks:

Each timer can be set to a maximum of 65535 seconds
Current timer values (using timer array facility) may be used in math expression as a variable.
The setting of a timer as no effect on system operation. The modules continue to operate as if that program line was never encountered.
Program execution is suspended when:
The WAIT FOR statement is encountered AND
The time specified has not elapsed.
The timer value may be checked during the running of a program by requesting the value of the timer array.
A timer may be set in a program or in Manual Control.
When the Zymate System is actively waiting for a timer to elapse, any actions taken by the operator are ignored.
If a program has to be aborted while it is waiting for a timer, it must be done by turning off the Controller’s power.
Several program statements may appear between the TIMER(x) = math expression and WAIT FOR TIMER x statements in a program. (This allows the system to continue operating while timers are active).

6.7.3 Examples :

Demonstration program for **looping** :

Name of the program : JOOCHAN.DEMO.LOOP
Description : this demonstration program takes a test tube from rack.1 and puts it into the evaporator station. This is done for 6 times .
After that, the opposite operation is performed : 6 test tubes are taken back to their former position in the rack .

Observation : indexes must be updated before exiting the loop . If not, wrong operations are performed .

```
display off
rack.1.index = 1
evaporator.index = 1
get.hand.a

do 6 times     - The loop is done 6 times

        get.from.rack.1
        put.into.evaporator
        rack.1.index=rack.1.index+1   - updating indexes
        evaporator.index=evaporator.index+1

enddo
rack.1.index=rack.1.index-1
evaporator.index=evaporator.index-1

do 6 times

        get.from.evaporator
        put.into.rack.1
        rack.1.index=rack.1.index-1
        evaporator.index=evaporator.index-1

  enddo

park.hand
rack.1.index=rack.1.index+1
evaporator.index=evaporator.index+1
```

Demonstration program for **pausing** :

Name of the program : JOOCHAN.PAUSE
Description : this demonstration program asks for the amount of time that the system is going to wait for . It is saved on a variable created by us named joochan.pause . During that time (in seconds) the system will do nothing but waiting .

```
display off
prompt Insert pause time
input joochan.pause
timer (1) = joochan.pause

wait for timer 1
```

Demonstration program for **condition (IF / THEN) :**

Name of the program : JOOCHAN.DEMO.IFTHEN
Description : this demonstration program takes a test tube from the centrifuge and takes it to the capper station. If the tube is capped (it is known by consulting the value of an internal variable named c:c.capped) then it is uncapped.If not, it is capped and is taken to rack.2.

```
1       display off
        get.hand.a
        get.from.centrifuge
        put.into.capper

10      if c:c.capped = 1 then 11 – when c:c.capped =1 the tube is capped
        cap
        goto 100
11      uncap
100     get.from.capper
        rack.2.index=1
        put.into.rack.2
        park.hand
```

Demonstration program for **inconditional jumping (GO TO) :**

Name of the program : JOOCHAN.DEMO.GO.TO
Description : this demonstration program asks for typing YES or NO, and its value is transferred to a variable created by us named joochan.time.

If the answer was Yes (= 1 ; it is the value stored in the variable called “yes” ;see Sample Data->View / Change Data ->Yes) the time is displayed (?clock). If not (No = 0), the program is finished by jumping to the end (code line number 100) .

```
1     display off
      prompt Write “YES” to see the time or “NO” to exit
      input joochan.time

10    if joochan.time = 0 then 100
      ? clock
      goto 1     - An inconditional jump to line 1 is executed

100
```

6.8 Reteaching positions when a station is placed into a different location :

For **hand.a :**

1. A dictionary must be already loaded .
2. Go to Module Setup -> Zymate II Robot
3. Delete Entry (press D)
4. Write the name of the station to be deleted (hand.a in this case)
5. Press Enter .
6. Next message will be displayed : Name deleted from Dictionary
7. Press R (RETurn)
8. Escape and come back to the Main Menu .
9. Insert the proper installation sofware diskette supplied by Zymark in the controller (in this case this disk is labelled as GP HAND.A) .
10. System -> Pyappend Dictionary .
11. Select the dictionary that appears and that deals thoroughly with the current station (for hand.A: cp900-1.zyd) .
12. Follow the instructions on the screen

    12.1. Select input target = number of the sector on the robot plate
    The new location chosed on the plate is on the 21st position around the Robot (former position on the plate was 46-47) .

    12.2. Press Y (yes) when asking for running Setup .
    12.3. Follow instructions
    12.4. When asked if the station mounted on a Pyplate, answer Y (Yes) .
    12.5. Hand.a has been placed successfully in another new location .

13. Press Enter .
14. Press Enter to run Test program and check successful configuration .

For **rack.1 :**

15. A dictionary must be already loaded .

16. Go to Module Setup -> Zymate II Robot
17. Delete Entry (press D)
18. Write the name of the station to be deleted (in this case : rack.1)
19. Press Enter
20. Next message will be displayed : Name deleted from Dictionary
21. Press R (RETurn)
22. Escape and come back to the Main Menu .
23. Insert the proper installation sofware diskette supplied by Zymark in the controller (in this case labelled as RACK 11 mm vial) .
24. System -> Pyappend Dictionary .
25. Select the dictionary that appears and that deals thoroughly with the current station (for rack.1 : Z013-1.zyd) .
26. Follow the instructions on the screen

    12.1. Select input target = number of the sector on the robot plate
    The new location chosed on the plate is on the 17-20th position around the Robot (former position on the plate was 4 to 7) .

    12.2. Press Y (yes) when asking for running Setup .
    26.3. Follow instructions
    26.4. When asked if the station mounted on a Pyplate, answer Y (Yes) .
    26.5. Run Test to check correct installation of the rack .
    12.5. Rack.1 has been placed successfully in another new location .

27. Press Enter .
28. Press Enter to run Test program and check successful configuration .

7. PROBLEMS & SOLUTIONS :

These are the main problems we have had during the execution of this project. After studying the system thoroughly, we succeded in making everything work without any external help. These are some solutions for different types of problems which might appear in the future .

- If the **Tumble mixer** does not work :

  1. Look if it is connected to the power !
  2. Look if the power cables work .
  3. Execute VORTEX.ON
  4. Look if the wires are properly connected and that they are not loose .

  Solution : fix the connector by pressing it to the controller while the vortex is ON until you hear the noise of the engine ON !

- Air **pressure** missing (on-screen error messages) :

  1. Look if there is air pressure connected to the PEC .
  2. Check if you have enough air pressure (minimum is 27 psi =185 Kpa) .
  3. Check if all the variables concerning air pressure are OK (i.e air.confirm.sensor) .

- **Balance** (on-screen warning messages) :

  1. Power ON the balance by pressing the front ON/OFF button
  2. Wait until it is stable.
  3. Check if all the pneumatic connections are OK .

- **Robot** (arm) :

  Problem : The robot is unable to open its fingers .
  Possible cause of failure : look at the arm of the robot, just where the hand is attached. Some pins are placed there. They transmit information . If the arm gets a hand with extreme speed, one of that pins is pushed backwards and electrical connections fail .

  Solution : draw any wrong pin to the right position (align it up) .

8. CONCLUSIONS :

After four months of intense work, this project is finished .
We have been the first people to study this system,as nobody at university knew nothing about it . Everything was completely new for us. Only with the manuals of the system and lots of hours spent on this, we have succeeded in making everything work.
First of all, all the equipment had to be assembled and connected . Afterwards, exhaustive study was carried out.
Later on, some demonstration programs were made and finally a general program was developed . Some stations were placed onto different locations, and new programming had to be done .

Meanwhile, some problems had to be solved, such as the lack of pessured air, some failures in two stations (the balance and the vortex station) and serious problems with a connection pin in the hand of the robot that made it not open its fingers .

As we have acquired vast experience in this field, we have also enclosed a User's Manual, so that next users of this equipment have proper and quick information of the robot and the different stations .

Finally, it is to say that all the aims have been achieved .

9 . REFERENCES

Zymark corporation
Hopkinton, MA 01748 USA.
Telephone (508) 435-9500 or fax (508) 435-3439

Zymark User's Manual, vol. 1 & 2
Diverse authors
Hopkinton, MA . Year of publication : 1988

www.zymark.com

Asea Brown Boveri (ABB). Sweden

www.abb.com

Seiko Instruments USA Inc .

Factory Automation Division
2990 West Lomita Blvd.
Torrance, CA 90505
Phone: (310) 517-7850 Fax: (310) 517-8158

www.seikorobots.com